12-1-1994

Logon to AI: A dynamic supporting tutorial system.

Carole L. Becic

LOGON TO AI:

A DYNAMIC SUPPORTING TUTORIAL SYSTEM

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

University of Nebraska at Omaha

by

Carole L. Becic

December 1994

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the requirements for the degree Master of Arts, University of Nebraska at Omaha.

Committee

Name	Department
Betty L. Hickman	Computer Science
Z. Cihan	Computer Science
David M. Sutherland	Biology

Chairperson _____

Date Nov. 15, 1994

ABSTRACT

This Master's Thesis presents a computer-aided tutorial system built on the concepts of dynamic support and on-line intelligence. The system was originally aimed at providing an interactive teaching aid for a college level introductory course in Artificial Intelligence (AI). The resulting system can actually be used for many other courses of various disciplines and at different educational levels. This can be done by simply supplying this system with the teaching materials of the particular courses.

The system is designed for easy use by both students and teachers. Students are guided by concise and friendly prompts along their journey through the message and question sessions. Teachers will find this system easy to tailor to his or her subject material and teaching style. The messages and questions are stored in individual files external to the program. These files are simple to create, easy to edit and can cover many different subjects.

The AI portion of this system is patterned after the textbook, Artificial Intelligence by Patrick Henry Winston. The material is carefully organized and presented in the system so that students may gain a better understanding of the principles of AI and command the key concepts of various AI subjects. A complete set of files, focused on Artificial Intelligence, was created to support the teaching of the fifteen major chapters of the textbook and to demonstration the tutorial system.

ACKNOWLEDGEMENTS

I would like to say thank you and express my deep appreciation to my family and friends. Especially my husband, Jim, for his constant encouragement, support, patience and love. A thank you goes to our children, Dan and Jennifer, who may not completely understand but will hopefully set and achieve their own goals. I would like to thank my friends, Sheri Zimmel and Anu Chebrolu, who have taught me a lot and have been very supportive. I wish to extend a special thank you to my best friend, Jane Marquardt, who has always been there for me with her own special kind of support and encouragement. I also wish to say heart-felt thank you to my parents who have always been encouraging and supportive in the various endeavours of my life.

I want to express my gratitude to Dr. Qiuming Zhu, my graduate advisor, and the members of my thesis committee, Dr. Zhengxin Chen, Dr. Betty Hickman and Dr. David Sutherland for their advice, patience and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The computer has become an intricate part of our lives - directly and indirectly influencing our lifestyles. Actually, it's the programs that run the computers that affect our lives. These programs come in many styles and types. Their influence can be found in nearly all facets of our lives from keeping track of our financial status to letting us communicate with our friends and family across the world.

In addition to industrial and business applications, computer programs are playing an increasingly important role in education. These educational programs are known as Computer-Aided Instruction (CAI) programs. They are designed in a wide variety of program styles to assist teachers both in and out of the classroom. CAIs can provide extra help for students without the personal supervision of a teacher.

The most common pedagogical assistance programs are the drill and practice programs. **Drill programs** use repeated activities to help students memorize facts which were previously taught elsewhere, in a classroom for instance. Facts, such as those for math or spelling, are presented repeatedly with hints or explanations for incorrect responses. Positive feedback is used to reinforce the student's correct answers. **Practice programs** teach skills, procedures, and processes previously learned elsewhere. Follow-up step-by-step instruction is embellished with positive and corrective feedback (Price, 1991). Drill and practice programs are used to "enhance retention and transfer" and to "strengthen already learned associations and to build skill in concept classification and rule using" (Wager and Gagné, 1988).

Page-turner programs, another type of CAI, contain static unchangeable printed material in a dynamic changeable format. While this may seem an inappropriate and inefficient use of the computer, the as-needed, easy to modify characteristic of the computer medium makes it

invaluable for on-line instruction manuals. A product change is easily compensated for in the manuals (Price, 1991).

Simulation CAIs are complex and powerful representations of real-life and imaginary events. Students can make decisions effecting the simulated environment and immediately and realistically experience the consequences of their decisions (Price, 1991; Wager and Gagné, 1988).

Some **Computer games** can also be considered a type of computer-aided instruction program. These games teach hand-eye coordination, problem-solving skills and the ability to follow directions. These side effects may be planned as they are in educational games or they may be "accidental by-products" of their entertainment capabilities (Price, 1991).

Simulation and educational systems can teach by allowing the student to learn by doing. These systems are less constraining than drill and practice software but give the student less "power" that general purpose languages. They "share the initiative with the user" by letting teachers and students make choices that "help frame the educational activity" (Riel et al., 1987).

Tutorial programs "are the most complete forms of CAI" (Wager and Gagné, 1988). These systems are designed to teach new skills by mimicking the "ideal" qualities (patience, good diction and liberally applied individual attention among others) of good human teachers and tutors (Ohlsson, 1987). These types of CAIs often include some learning guidance, reviewing, testing and practice (Wager and Gagné, 1988). The material is usually presented in text form but may include pictures, graphics, sound and/or animation to emphasize and clarify.

Most intelligent tutorial systems present new subject material in small amounts. The students are asked questions about the material presented and their answers are evaluated immediately. Incorrect student responses are handled without delay, using "correction feedback", by

repeating the relevant material giving hints or clues to help clarify the material. Correct answers are "positively reinforced" with praise and often by the repetition or restatement of the correct answer (Price, 1991).

Some tutorial programs provide drill and practice segments as well as summaries to reinforce and reemphasize the material. Evaluation and diagnostic testing segments may be included to track the student's progress, allowing segments of material that he or she has already mastered to be skipped.

A **linear tutorial program** is one type of intelligent tutorial program. Even though the instruction sequence may be repeated, these programs follow a definite sequence of instruction (Yazdani, 1987). Linear tutorial programs adapt to the individual student only through their self-pacing feature and when multiple attempts to answer questions are allowed (Price, 1991).

Branching tutorial programs adapt much better to the individual student. They provide problems of appropriate difficulty for individual students. Corrective feedback is provided as the programs adapt to student responses (Yazdani, 1987). These programs have diverging paths to help students having trouble grasping particular concepts. These paths are never seen by students who have mastered the concepts. Some programs let the student decide which portion of the material to study. Other programs may also have pretesting branches which allow the student to skip the material he or she already understands (Price, 1991).

Riel, Levin, and Miller-Souviney (1987) propose a process called **"dynamic support."** Dynamic support provides the student with a great amount of help and support initially in learning a new topic. As the student gains increasing expertise, support is gradually reduced until the student is an expert on the subject (Riel et al., 1987).

A tutorial system using dynamic support mimics a human teacher by presenting new skills and information in small amounts, adding new material to what the student already knows and has mastered (Winston, 1991; O'Malley, 1990). As a skill is learned, the system is able to gradually withdraw its support and guidance until the student is an expert.

The ideal learning environment for a student is to have the full attention of a human tutor who is willing and able to adapt his or her teaching techniques as the student's learning needs change (Ohlsson, 1987). Students can ask questions as they go along. The teacher is able to respond immediately, clarifying misunderstandings and reinforcing difficult concepts with repetition and/or a restatement of the concept. The student's progress and understanding is constantly monitored, allowing the teacher to advance at the student's learning pace. This one-on-one environment lets the student learn at their own pace (Burns and Parlett, 1991). This idea is also supported by Ohlsson who states that "...the main promise of computer tutors, I claim, lies in their potential for moment-by-moment adaptation of instructional content and form to the changing cognitive needs of the individual learner,..." (Ohlsson, 1987). This is the environment strived for by designers of these Intelligent Tutorial Systems (ITSs).

There are already sophisticated intelligent tutorial systems developed and being researched. For instance, LISP teaches the computer language LISP through an interactive program which emulates "some of the advantages of a personal tutor" (Corbett and Anderson, 1992). PROUST (Sack and Soloway, 1991) was built to interactively teach the computer language Pascal. CHIRON is a descendent of PROUST which includes an improved interface and a refocused error finding emphasis (Sack and Soloway, 1991).

There is, of course, much work to be done. New ideas are constantly being development. Cumming and Self (1989) proposed adding a discussion

level of comments and advice to the simulations of an intelligent educational system. The learner would then be "...considered as a collaborator in learning, rather than offering instruction to the user, considered as pupil" (Cumming and Self, 1989). Correcting inadequacies in existing systems (For example PROUST's inability to allow students to ask questions, which were remedied in CHIRON (Sack and Soloway, 1991).) are also being accomplished. The resulting intelligent tutorial systems will more closely emulate human teachers and, hence, will be invaluable as teaching aids.

Table I was compiled from R. V. Price (1991) with examples of each type of CAI mentioned above.

TABLE I Computer-Aided Instruction Programs

TITLE	TYPE	PUBLISHER	COMMENTS
"Spelling Tutor"	Drill	Sunbelt Technologies	Uses the flash card concept
"The Food Processor"	Practice	Cambridge Development Laboratory	To practice planning healthy meals
"LEARN"	Page-turner	Digital Equipment Corporation	Provides on-line VAX main-frame information
"Where In The World Is Carmen San Diego?"	Simulation	Broderbund Software Co.	Teaches language skills, problem-solving, social studies
"Reader Rabbit and the Fabulous Word Factory"	Educational Game	The Learning Company	For young beginning readers
CHIRON	Tutorial	Sack and Soloway (1991)	Teaches Pascal Programming

This document explores the creation of the tutorial system, *An Introduction To Artificial Intelligence*. As a summary, the various section

topics will be described. The section SYSTEM OBJECTIVES describes the various goals set for this project and the method followed to achieve each goal. SYSTEM DESCRIPTION explores the general workings of this tutorial system. FEATURES AND METHODS points out the important considerations in this tutorial system and how they were executed. The section IMPLEMENTATION describes the files supporting this system. The file structure, the importance of the file naming method and the purpose of the filenames are detailed. The section THE COMPLETE PACKAGE describes the thinking and reasoning that went into the creation of the supporting files (These are the message, extra-help message, question, extra-help question and the menu, introduction and title files.) which contain the subject material related to teaching an introductory college-level course in Artificial Intelligence. FUTURE WORK suggests features and improvements which could be appended this system to enhance its teaching value. Several sample tutorial sessions are contained in the section SAMPLE TUTORING SESSIONS. The APPENDIX contains all of the files created to complete this tutorial system on Artificial Intelligence.

SYSTEM OBJECTIVES

The first objective of this project was to produce a software package that could be learned quickly and used easily by both teachers and students (Nelson, 1993). The idea was to construct a system which required little or no programming experience to use and understand it.

The attempt to achieve this objective was made by keeping the software simple and straightforward for both the teacher and the student. The student is guided through every step of a tutoring session by short uncomplicated prompts. Brief clear messages and unambiguous questions appear centered on neatly bordered screens. A student responds to the questions by simply pressing a single-key. The teacher creates the messages and questions in his or her own teaching style. They are stored in uncomplicated formats in individual files external to the program. These individual files, for the desired topic, can then be easily created using an familiar editor or word processing software package.

The second objective was to make the program general enough to be used by instructors of many different subjects, to make it domain-independent (O'Neil et al., 1991; Lawler and Yazdani, 1987). This objective is part of the description of an unintelligent tutoring system. The unintelligent tutoring system "knows nothing of the specific problems under consideration and has no model of the students that use it (Nathan et al, 1989)."[1] Domain-independence seemed important to include in this project in order for the software package to be used in subject areas other than Artificial Intelligence. Domain-independence was achieved here by separating the subject material from the program. Separate files hold the

---

[1] This tutorial system is intelligent in that it responds to the student's answers by providing further explanation and questions when needed.

subject-oriented introductions, messages and questions. The program drives the presentation of the messages, questions and the extra-help information.

The third project objective was concerned with flexibility for the instructor. The use of separate files also facilitates editing flexibility by allowing individual file creation, editing and the manipulation of information as the need arises. The need to alter the main program is then eliminated.

Teaching flexibility is also included. The questions and extra-help questions can be newly created by the teacher, obtained from a textbook, publisher-supplied or gleaned from any other question pools. Answer files may contain short messages specifically to reinforce or clarify any new information provided by the questions. The information messages and extra-help messages can be tailored to the individual student or to the general characteristics of a particular class. All messages may be eliminated entirely if the system is to be used for testing purposes only. The messages may be gentle reminders encouraging the student to think about the subject material before answering or they may be intricately detailed to do most or all of the teaching. The extra-help questions need not be used if drill and practice is all that is required.

The fourth project objective was to make the program unintimidating (friendly) for the student. One way to achieve this goal was to create comfortable interfaces. Bordered screens present a more finished, more appealing appearance. The various border colors relieve the harshness of the white letters on a black background. (Backgrounds of other colors reduced the emphasis on the text.) The various border colors were chosen with care to emphasize the importance of the messages and reduce the implied stress of answering the questions (Salomon, 1993).

The second method to make the system less intimidating was to create positive, encouraging, and friendly program responses to the student's

answers. These responses must be short and to the point (Nelson, 1993) allowing the student to move quickly from question to question thus eliminating frustration. The messages following incorrectly answered questions were designed with a hopeful positive tone.

The student is also given some control over the tutoring session. The session progresses at the student's own learning pace with the opportunity to exit the program given frequently. The student can relax knowing he or she has control of the session.

SYSTEM DESCRIPTION

This tutorial system gives the student the ability to choose a study topic through a system of menus. The student begins by choosing a general topic[2] from the main menu. A brief summary highlights the main ideas and introduces the chosen topic to the student.

The subject area is further narrowed using a second layer of menus which specifically pertains to the general topics of the main menu. A chapter topic is chosen from a second menu and introduced through a brief summary in the next screen. The menus allow the student to skip the material he or she already knows (Price, 1991).

At this point, brief informative messages present the information covered within the chapter. The messages may be written generally, to gently prompt or guide the student into thinking about subject material already taught elsewhere. More intricate messages may be written in great detail to explain or actually to teach the information. The decision to use general or detailed messages is left to the teacher. This feature helps the teacher mold the tutorial system to his or her own teaching needs and is supported by Riel, Levin, and Miller-Souviney (1987). It also supports the flexibility objective mentioned earlier.

Each message is followed by one or more true/false, yes/no[3] or multiple-choice questions. The questions are created and organized by the teacher with each question pertaining to the most recently presented message. The student is prompted to respond with an appropriate yes or no,

---

[2] The word "part" is specific to the textbook Artificial Intelligence (Winston, 1992) from which subject material for the messages and question contents was drawn. "Topic" is a more general designation that can be applied to any subject material covered. "Topic" and "Part" are interchangeable in this document.

[3] The yes/no questions were included to add variety and allow questions which are better answered by yes or no rather than a true or false.

true or false or letter for each type of question. An invalid answer, handled by asking the student to re-enter the answer.

A dynamic support structure intelligently guides the tutorial system along a path that changes with the correctness of the student's answer. The student's answer is immediately evaluated. A correct answer receives a brief congratulatory message. An incorrect answer produces a response that notes the incorrectness of the answer but attaches no stigma to the failure (Schank and Edelson, 1989). A wrong answer is handled according to the question's type. If the question is a multiple-choice, the student is given a second chance to answer the question. The second chance answer is evaluated in the same manner as the primary answer and an appropriate message is printed according to the answer's correctness.

The true/false and yes/no questions are handled somewhat differently. Giving the student a second chance to answer these types of questions makes no sense. Instead, the student is informed that the answer is not correct and, through a brief message is asked to reconsider his or her answer. The correct answer is then presented along with any message accompanying the answer.

An incorrectly answered primary question always leads to the extra-help portion of this tutorial system even when the second chance answer is correct. An extra-help message is printed to help clarify the correct answer or to restate the previous message. An extra-help question (or a series of questions) is (are) asked of the student. The extra-help question(s) are handled the same way the regular questions are except there are no extra-help messages or follow-up questions. The answers are evaluated and the appropriate messages are applied.

At the end of a chapter of questions, the student is congratulated on his or her completion of the chapter. A message asks if the student

wishes to continue. If so, the program returns to the main menu to begin again.

The decision to end a tutoring session is frequently presented, allowing the student some control over the program. The simple prompt "Next Question? (y/n)" allows the student to continue to the next question or to end the program. The prompt appears after the evaluation of both the primary and extra-help question answers. The decision to end the program is also included as an option in each menu.

Figure 1 contains a flow chart illustrating the progression of the system through a tutorial session. Several sample tutoring sessions can be found in the section "Sample Tutoring Sessions".

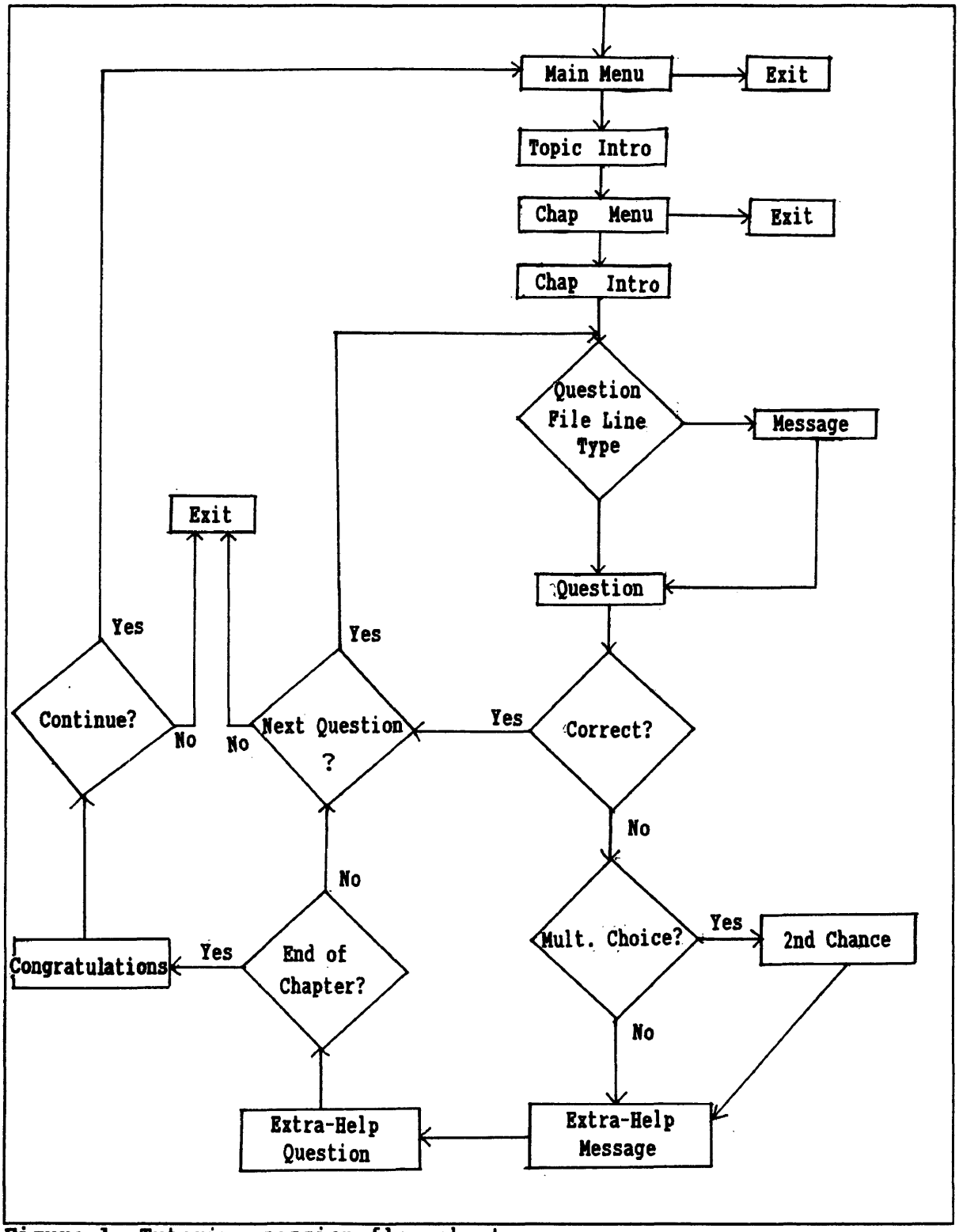


Figure 1 Tutoring session flow chart.

FEATURES AND CONTRIBUTIONS

Authors and designers agree that the interface of an Intelligent Tutoring System is very important in the learning process of the student (Bonar, 1991; Burns and Parlett, 1991; Chabay and Sherwood, 1992; O'Malley, 1990). "The power of an educational tool lies in the user interface and its ability to provide meaningful cues which trigger the appropriate reasoning from the student (Nathan et. al., 1989)."

The computer displays are the most obvious and may be the most important parts of this system (Nathan, et.al., 1989). A great deal of thought went into adapting the limited resources available into a effective interface. The displays of this system are kept as uncluttered as possible to hold the student's attention and avoid overwhelming him or her with too much information at one time. Blank space between the borders and the text helps focus the student's attention to the messages and questions presented as is described by Chabay and Sherwood(1992). Capitalization emphasizes important words and concepts within the messages.

Salomon (1993) suggests creating a learning environment by using colors. An uncluttered environment is created here by using colored borders that change to emphasize the screen's content.

To further reduce clutter, the questions are displayed on screens separate from the messages. The questions remain on display during the answering, evaluation and the second chance portions of the session. This allows the student to reconsider and compare his or her answer with the question. The incorrect answer remains on the screen during the student's second chance at answering correctly to avoid repeating the first answer. Short clear system prompts guide the student to his or her next response.

This tutoring system provides the student with some control over the tutoring sessions, an important learning feature described by Kearsley (1988). Through the menus, the student is able to choose the topics he or

she wants or needs to explore. Subjects already mastered, can be skipped by choice through the menu system. The ability to control the session's length (The "Next Question?" prompt.) is frequently encountered, giving the student ample opportunity to continue or end the current session.

The main contribution of this tutorial system is the incorporation of dynamic support into the message and question tutoring sessions. Dynamic support provides immediate on-line evaluation of the answer, thus reinforcing correct answers. A student is given the opportunity to reconsider an incorrect answer and to try again to answer the question correctly. The system is also dynamic in that it adapts to the student's answers by branching to the extra-help portion of the program when necessary. This dynamic adaptability generates different tutoring sessions directed by the student's knowledge of the subject. These features mimic the characteristics of an ideal human teacher or tutor (Ohlsson, 1987; Burns and Parlett, 1991).

Chabay and Sherwood (1992) have described the importance of allowing a student to learn at his or her own rate. A student is better able to learn when he or she has control over a system's teaching rate and is able to regulate it to his or her own personal learning speed. Self-pacing is strongly incorporated into this system by allowing each screen of information, message or question, to remain until the student presses a key. The student is thus able to decide the speed of each session. A complete session may be repeated if the student wishes further study or the teacher deems it necessary (Chabay and Sherwood, 1992).

This tutorial system also contributes to the domain of educational software by filling the void that surrounds the subject of Artificial Intelligence. This system provides an intelligent tutorial system specifically for teaching Artificial Intelligence. A system such as this has not been available until now. It can be used as a ready-made support

device by using the message and question files, pertaining to AI, included in this software package. A hard-copy of these files is available in the appendix.

Another contribution made by this system is that when the system is isolated from the subject-oriented files (introductions, messages, questions, etc.) it is actually a driver program. This feature allows it to adapt to many subjects and educational levels. Teachers from grade school through college can use this tutorial system by including subject specific messages and questions geared to the participating student's age and grade-level.

IMPLEMENTATION

This intelligent tutorial system was written on an IBM Compatible 386 SX computer. The computer uses *DOS 6.0* as its operating system. The source code was written in C++ using *Borland's C++, Version 2.0*.

The tutorial system requires 91 Kbytes of memory to run. There are 120 support files (messages, questions, etc.) for *An Introduction To Artificial Intelligence* which require 128 Kbytes of storage space.

File Structure

To facilitate the implementation of the dynamic support structure, this tutorial system uses external files to store the subject-related material. These files are categorized into different types to accommodate the messages, questions, menus and a variety of other contents. The separate files allow a wider variety of subjects to be taught and gives the instructor greater editing flexibility.

The file structures have been kept as uncomplicated as possible in consideration of the nature of the potential users of this program. The files may be created in a word processing software package or with any editor with which the instructor is familiar. The files must be saved as "DOS text" or "generic text"[4], in the directory called "*aitutor*".

All files have some shared characteristics. All files use four asterisks ("*****") to signal the end of the file. The area of the computer screen set aside for text is 15 lines long (2.5 inches) by 50 to 55 characters wide (5.0 to 5.5 inches) and is centered. Hence, the introduction, message and question files should abide by these parameters as closely as possible for the best looking presentation.

---

[4] These two terms are text saving terms used by WordPerfect 5.1.

What's In A Filename?

During the development of this tutorial system, it became apparent that the filenames should be used to organize the directory containing the system's files. Thus, much consideration went into the creation of the filenames. A clear and easy file identification system involving the filename was created because of the large number of files involved with the system, all of which are stored in the directory "*aitutor*".

Each chapter has individual files associated with it which contain the chapter's introduction, questions, answers and messages. All chapter files have names in the form of "pxcy.ext". The "p" and "c" are constant and stand for "Part"[5] and Chapter." The "y" is chapter's number and "x" is the number of the topic (or part) where the chapter is located. As an example, the filename "p3c29." indicates a file pertaining to Chapter 29 which is included in Part 3.

The filename extensions designate a file's contents. For example, the file extension - .exm - contains extra-help messages. Figure 2 contains a list of the extensions and their meanings.

.int	Chapter Introductions
.q	Main Question Files
.a	Main Answer Files
.m	Main Message Files
.exq	Extra-Help Questions
.exa	Extra-Help Answers
.exm	Extra-Help Messages

Figure 2 Filename extensions.

Other content specific files are included in the *aitutor* directory. Filenames of the form "menuz.txt" contain the menu text for each topic found in the main menu. "z" indicates the topic's menu number. The main

[5] "Part" and "Topic" are interchangeable. Please note footnote 2.

menu text is found in the file called "mainmenu.txt". An illustration of the complete filename for each file type is located in Figure 3.

File Type	Filename
Menu	menu1.txt
Introduction	p1c7.int
Message	p1c7.m
Question	p1c7.q
Answer	p1c7.a
Extra-Help Message	p1c7.exm
Extra-Help Question	p1c7.exq
Extra-Help Answer	p1c7.exa

Figure 3 Filename examples.

Question and Answer Files

The question files (.q files) are the most important of the chapter files. Special symbols, letters and numbers found at the beginning of some text lines, tell the program what to do or what question is found in each line.

Messages are indicated by the symbol "^" followed by the number of the message to be presented to the student. A message should start each question file. The messages may be incorporated as frequently as the instructor wishes or they may be nonexistent.

Questions are preceded by a number and a letter separated by a space. The number is the question number which is used to find the question's answer in the answer file. The letter indicates the question's type - true/false (t), yes/no (y) or multiple-choice (m) - and tells the program how to handle the question. Figure 4 lists the question types and their letter designations.

```
Multiple-Choice :  M
True / False    :  T
Yes / No        :  Y
```

Figure 4 Question types.

Each question is followed by a percent symbol ("%") and a number indicating the extra-help message to be printed if the question is answered incorrectly.

The question and extra-help question files are handled identically. The only difference is that the extra-help questions have neither extra-help messages nor extra-help questions of their own. The Appendix contains all of the question and extra-help question files for this tutorial system. Figure 5 and Figure 6 contain brief examples of a question file and extra-help question file respectively.

```
^1
1 m A representation using nodes to denote paths
    and branches or links to show path extensions,
    is called a:
    a.  Search Map
    b.  Flow Diagram
    c.  Search Tree
    d.  Path Search
%1
2 t A partial path in a search tree is a path
    which goes to a node with no branches leading
    away from it.
%2
****
```

Figure 5 A question file example.

```
^1
1 3 A search tree is often used to arrange all
    possible paths from one place, the start
    node, to another, the goal node.
^2
2 1 Jane used a search tree to discover all the
    routes from Omaha to Lincoln.  She found one
    path which began in Omaha and ended in Sioux
    City.  Is this path a
    a.  Branching Path  or
    b.  Partial Path    or
    c.  Scenic Path     or
    d.  Complete Path
****
```

Figure 6 Extra-help question file example.

The answer file (.a) and extra-help answer file (.exa) files are simply and identically arranged. The question number is followed by the question's correct answer. Multiple-choice question answers (a single letter) may be upper or lower case letters. True/false and yes/no answers must be typed out but may be in all upper, all lower case or may have only the first letter capitalized. If a short explanation is to be included it is typed on the next line immediately following the answer. A short explanation is ideal for the extra-help questions which lack extra-help support for themselves. Figure 7 shows a portion of an extra-help answer file including a short explanation as an example. The complete set of answer and extra-help answer files for this tutoring system are included in the Appendix.

```
 1 yes
 2 b
 3 true
 4 false
       Depth-first travels down a tree before it
     travels across.  Thus, visiting a leaf node
     before the root's second child node.
 5 yes
****
```

Figure 7 Answer & extra-help answer file examples.

Message Files

The message and extra-help message files are also identical in structure. Each message is preceded by the message's identification number and separated from the next message by four plus symbols ("++++"). Capitalization can be used to emphasize important words and phrases (Chabay and Sherwood, 1992).

If a message is longer than 15 lines, a series of four "####" is placed on the 16th line to indicate a page break. One-paragraph one-screen messages are best, however, to avoid overwhelming or boring the student (Chabay and Sherwood, 1992). To center the messages within the left and right screen borders, the lines should be from 50 to 55 characters, about 5.25 inches. Figure 8 illustrates message and extra-help message files with a section of an extra-help message file. The complete set of message and extra-help message files is contained in the Appendix.

```
++++
4    BREATH-FIRST SEARCH visits all nodes on a
particular tree level before it randomly extends
the path one step to the next lower level of the
tree.
++++
5    A NONDETERMINISTIC SEARCH randomly chooses
a node to expand.  It keeps the search from
exploring too many branches or too many levels.
It is a middle of the road search between
depth-first and breadth-first searches.
****
```

Figure 8 A message file example.

Miscellaneous Files

Several addition files were incorporated into the system to increase its subject material adaptability. The files are the title file, the menu files and the introduction files.

The title file, title.txt, contains the 4 lines of the title screen. The four lines are typed straight into the title.txt file. The program automatically centers and spaces each line on the screen. This file is changed when the subject matter being taught is changed. Figure 9 contains the title.txt file for this tutorial system.

```
AN INTRODUCTION TO ARTIFICIAL INTELLIGENCE
A Tutorial System
Written by
Carole L. Becic
*****
```

Figure 9 The title file.

The menu files (mainmenu.txt and menu_.txt) are entered as they are to appear on the screen with the title lines preceded by a dash ("-"). A

total of 15 lines of 50 to 55 characters may be entered. Figure 10 contains an example of a menu file.

```
-PART II:
-Learning and
-Regularity Recognition

16.  Learning By Analyzing Differences
22.  Learning By Training Neural Nets
     Press 0 to exit program
****
```

Figure 10 A menu file example.

An introduction file is also created for each main menu topic (Part)[6] " menux.int" and for each Chapter "pxcy.int". These files are constructed in the same manner as the menu files. The introduction files contain brief summaries of what is covered in each Part or Chapter. Figure 11 shows, as an example, the introduction file of PART 3. The complete set of introduction files has been placed in the Appendix.

```
-PART III
-Vision and Language

     Part III introduces you to visual perception
and language understanding.  You will learn  about
recognizing objects, describing images, language
constraints and how to express them, and about
responding to questions and commands.
****
```

Figure 11 An introduction file example.

[6] Please see footnote 2.

THE COMPLETE PACKAGE

A very large portion of this thesis project was concerned with the creation of a full compliment of files to fill-out and complete the tutorial system. It also serves as a demonstration. The resulting software package is titled "An Introduction to Artificial Intelligence, A Tutorial System". It covers the fifteen major chapters of the textbook *Artificial Intelligence* (Winston, 1992).

Each chapter has a full set of files - an introduction file, message and extra-help message files, question and extra-help question files, and the answer and extra-help answer files. Files are also included for the menus, the introductions and the text of the title screen. All introductions, messages and questions were newly and originally created. No question pools or any other question sources were used or consulted.

Great care was taken to duplicate all of the work a teacher would need to perform in constructing the files for an effective tutorial system with the dynamic support features. The file construction was undertaken from a teacher's perspective. Several important questions considered were: (1) "Is it easy to create and edit the files?", (2) "Are there too many symbols to learn, thus making the files too complicated to easily build?", (3) "Does file construction and the files themselves make sense?". The entire file system was objectively evaluated and modified accordingly.

Creating the files for a chapter averaged eight to ten hours. The length of this time period depended on how difficult the subject was, how detailed the questions and messages were, and the number of questions and messages needed to cover the material efficiently. Although this length of time may seem extreme, carefully worded messages and questions are important to the effectiveness of the system. Thus, the effort is worth the time. An instructor, of course, would not need to compose the entire

system of files at one sitting. Individual chapters can be added into the system as a semester progresses.

The Messages

Composing the messages, questions and extra-help messages and questions proved to be the most challenging and difficult part of the project. The chapters were carefully read and meticulously scrutinized to determine the best wording to insure that the messages are unambiguous, clear, to-the-point and not too long.

The messages were composed around the important topics covered in each chapter of the textbook. Great care was taken to avoid repeating the textbook verbatim. Not only is that plagiarism but, from personal experience, an idea or concept may be better understood if it is presented in more than one context. The repetition of redescribing a subject helps to reinforce the idea in a student's mind (Price, 1991).

Many of the messages used merely mention the topic rather than going into great detail. This technique was often used to avoid "spoon feeding" the student. It seemed better to lead the student into thinking about the subject rather than supplying the exact answers for the question in the preceding message. Wager and Gagné (1988) referred to this as using their "constructive memory rather than reproductive recall." Merely mentioning the message topic was also used when the subject was too narrow to create more than one uniquely worded message. Wording the extra-help messages identically to the primary was always avoided.

Some messages were composed to explain individual parts of a larger concept or process. Chapter 13, for example, has a message explaining each of the nine steps to transforming a complex logical expression into its clause form.

Examples are included in many messages to help clarify and reinforce an idea. Often pictures, diagrams and examples explain a concept better than words (Chabay, 1992). Chapter 13's extra-help message number 5, for instance, contains a logic expression and the truth table that proves it. Extra-help message number seven in Chapter 7 has an example of back chaining to help clarify the process.

There were several times when an extra-help message was appropriately applied to two separate primary questions. This situation occurs in the third and fourth questions for Chapter 4 and again in Chapter 7 for the first and second questions. In both examples the two questions pertain to different aspects of the same subject and message.

All of the messages and extra-help messages for this tutorial system can be found in the Appendix.

The Questions

The questions also must be unambiguous and clear. Other factors were also considered such as: (1) Does the question pertain only to the previous message? If not, does the question use subject material not yet covered? (2) What about variety? Are there too many multiple-choice questions asked in a row? (3) Are the correct answers varied enough to avoid forming a pattern? (Are there too many "c" choices?)

The questions written for this system were composed mostly by employing personal experience gained from being a student. Wager and Gagné (1988) include asking questions in the sixth event of their nine events of instruction. In Event 6, questions are employed to elicit performance (an answer) from the learner. Event 7 then provides feedback in the form of "knowledge of the correct response" or presenting the correct answer if an incorrect answer is given (Wager and Gagné, 1988).

Three general types of questions - true/false, yes/no and multiple choice - were used. The true/false and yes/no questions are handled in the same manner. The yes/no questions were added for variety in consideration of the potential student user's attitude.

Variety was included to help keep the student's attention and reduce boredom. The fill-in-the-blank style of question was incorporated into some of the multiple choice questions. Other questions concern simple diagrams included on the screen with the question. Some questions involved having the student work out the problem before choosing an answer. Extra-help question number five in Chapter 13 requires working the truth tables of the expressions in the answer choices before finding the match to the truth table in the question.

The complete set of questions and extra-help questions can also be found in the Appendix.

SAMPLE TUTORING SESSIONS

The following pages contain the text portion of two short tutoring sessions. The menu, introduction, message, question and incidental intermediate screens have been included.

To emphasize the different parts:

- the student responses appear in **bold**
- the tutoring system's responses appear in *italic*
- lines which overwrite the positions of other lines are underlined
- individual screens are separated by "+++++++++".

Session I

```
++++++++++++++++++++++++++++++++++++++++
           AN INTRODUCTION TO

         ARTIFICIAL INTELLIGENCE

              Written by
           Carole L. Becic

       Press any key to continue.
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++

         AN INTRODUCTION TO
       ARTIFICIAL INTELLIGENCE

Part 1:  Representations and Methods
Part 2:  Learning and Regularity Recognition
Part 3:  Vision and Language
         Press 0 to exit program

 Choose a PART by entering its number ->  1
++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                    PART I
           Representations and Methods

     Part I introduces you to Artificial Intelligence
as a whole.  You will learn about the representations
and methods used to explain various types of intelli-
gence.  Part I teaches you about problem-solving methods,
search methods, rules and rule chaining, constraints
and propagation and more.

             Press any key to continue.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++++++++++++++++++++++++++++++

                PART 1:
        Representations and Methods

 1.  The Intelligent Computer
 2.  Semantic Nets and Description Matching
 3.  Generate and Test, Means-Ends Analysis
       and Problem Reduction
 4.  Nets and Basic Search
 5.  Nets and Optimal Search
 6.  Trees and Adversarial Search
 7.  Rules and Rule Chaining
 9.  Frames and Inheritance
13.  Logic and Resolution Proof
     Press 0 to exit program


Choose a Topic by entering its number ->  2

++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                    CHAPTER 2
        Semantic Nets and Description Matching

     Chapter 2 teaches you the importance of a good
representation.  You will learn how to gage the quality
of a representation.  Chapter 2 will give you a good
understanding of semantic nets and the important problem-
solving method called "describe and match".  You will
also be introduced to a geometric-analogy problem solver,
a plot recognizer and an object identifier based on
feature recognition.


              Press any key to continue.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   A REPRESENTATION is a set of conventions which
describe a class of things.


              Press any key to continue.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

A good representation must:
a.  Clearly define the problem's objects,
    relationships and inherent constraints.
b.  Present only relevant important details.
c.  Presents the problem simply, concisely and
      completely.
d.  All of the above.

Your answer? ->  d


                  Correct!

           Next Question? (y/n)  y

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

++

A DESCRIPTION uses the conventions of a representation to describe a specific thing precisely and clearly.

Press any key to continue.

++

++

The node-and-link representation is good to use for the farmer, fox and goose problem because it:

a. Explicitly describes the important objects and their relations.
b. Clearly presents the problem's inherent constraints.
c. Is easy to understand and easy to find the problem's solution.
d. All of the above.

Your answer? -> **a**

Your answer is not correct.

Please try again -> **c**

Your answer is still not correct.

The correct answer is : **d**

Press any key to continue.

++

++

Next Question? (y/n) **y**

++

++

Let's look at this topic another way.

Press any key to continue.

++

++

A DESCRIPTION uses the sets of rules and methods of a REPRESENTATION to describe a particular object within the representation's group of objects. For instance, a description might present a Lamborgine as a particular object in the group in the representation of "cars".

Press any key to continue.

++

++

A good description:
a. Clearly defines a particular object within a group.
b. Contains no irrelevant or redundant characteristics.
c. Must describe a particular group of objects.
d. Answers a and b.

Your answer? -> **d**

Correct!

Next Question? (y/n) **y**

++

++

A REPRESENTATION has four main parts:
- The LEXICAL part describes the vocabulary of the representation, the symbols used.
- The STRUCTURAL part tells the way the symbols may be presented, the constraints on the arrangement of the symbols.
- The PROCEDURAL part has the procedures to allow manipulation of the descriptions, their creation, modification and answering questions using them.
- The SEMANTIC part creates a method of associating the descriptions and meaning.

Press any key to continue.

++

++

Lexically a semantic net has constructor procedures to create the objects and links used.

Your answer? (True or False) -> **true**

Your answer is not correct.

Please reconsider your answer.

Then press any key to see the correct answer.

The correct answer is : false

Press any key when you are ready.

Next Question? (y/n) **n**

++

++

All Done For Now

Good-bye!

Press Any Key To End

++

Session II

```
++++++++++++++++++++++++++++++++++++++++++++++

          AN INTRODUCTION TO
       ARTIFICIAL INTELLIGENCE

             Written by
          Carole L. Becic


       Press any key to continue.

++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++

         AN INTRODUCTION TO
      ARTIFICIAL INTELLIGENCE


PART 1:  Representations and Methods
PART 2:  Learning and Regularity
           Recognition
PART 3:  Vision and Language
         Press 0 to exit program

Choose a PART by entering its number ->  3

++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    PART III
              Vision and Language

     Part III introduces you to visual perception and
language understanding.  You will learn about recog-
nizing objects, describing images, language constraints
and how to express them, and about responding to ques-
tions and commands.


           Press any key to continue.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++

          PART III:
    Vision and Languages

26.  Recognizing Objects
27.  Describing Images
28.  Expressing Language Constraints
29.  Responding To Questions and Commands
     Press 0 to exit program

Choose a Topic by entering its number ->  29

++++++++++++++++++++++++++++++++++++++++++++++
```

++

CHAPTER 29
Responding To
Questions and Commands

In Chapter 29 you will learn how WORD-ORDER REGULARITY can be used to turn English questions or commands into RELATIONAL DATABASE COMMANDS. You will also learn about SYNTACTIC TRANSITION-NET GRAMMAR, NESTED-BOX DIAGRAMS and SEMANTIC TRANSITION-TREE GRAMMARS.

Press any key to continue.
++

++

A GRAMMAR is viewed, in AI, as a representation. It is a set of conventions for "capturing linguistic knowledge". A SYNTACTIC TRANSITION-NET GRAMMAR is a sentence net and a set of supporting nets.

Press any key to continue.
++

++

A syntactic transition-net is
a. Used only to identify noun phrases.
b. Used to test word groupings to determine if they are valid sentences.
c. Made up of a sentence net and a set of supporting nets.
d. Answers b and c.

Your answer? -> **a**

Your answer is not correct.

Please try again -> **d**
Well done!

Your second try is correct!

Next Question? (y/n) **y**

++

++

Let's look at this topic another way.

Press any key to continue.
++

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    A SYNTACTIC TRANSITION-NET is used to determine
if a group of words is a valid sentence.  The process
involves starting with the sentence net then moving
to the noun-phrase net, the verb-phrase net and the
prepositional-phrase net as each link in the sentence
and supporting nets indicate.

           Press any key to continue.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Using this simple grammar:
      Noun phrase  verb phrase
     O------------>O----------->end
     NP determiner   noun       __
     O------------>O------>end  |
                           |<--
     VP verb    noun phrase
     O------->O------------->end
     PP preposition   noun phrase
     O------------->O------------->end

     Is this a sentence?
     "The beautiful weather turned bitter cold."


Your answer? (Yes or No) ->  yes

                  Correct!

            Next Question? (y/n)  y

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    A group of words is a valid sentence if the sentence
net and all of the supporting nets and all of the their
links are traversed.

Your answer? (Yes or No) ->  Y

            Please reconsider your answer.

        Then press any key to see the correct answer.
              The correct answer is :  no

           Press any key when you are ready.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                  Next Question? (y/n)[7]  y

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

[7] This prompt is moved to its own screen to prevent printing it too far down on the previous screen. This treatment leaves the important lines on the screen while the student needs the information rather overwriting the lines.

++

Let's look at this topic another way.

Press any key to continue.

++

++

A group of words may be a valid sentence without traversing all of the supportive nets. A sentence, for instance, may not have a prepositional phrase or an adjective. The sentence, "The goat runs." is such a sentence.

Press any key to continue.

++

++

A sentence is proven valid when the sentence net is completely traversed but not necessarily all of the supportive phrase nets have been traversed.

Your answer? (Yes or No) -> **y**

Correct!

Next Question? (y/n) **y**

++

++

A NESTED-BOX DIAGRAM is used to graphically illustrate what takes place as a route is traveled through a grammar. Each box represents a link in the syntactic transition-net. As each link is accepted or rejected, the action is so noted in the box.

Press any key to continue.

++

++

A nested-box diagram, for a particular group of words, graphically shows the path followed through a grammar to prove or disprove that it is a valid sentence.

Your answer? (True or False) -> **true**

Correct!

Next Question? (y/n) **yes**

++

++

Some sentences can be translated into RELATIONAL DATABASE COMMANDS. Relational databases hold information in tables of rows and columns.

Press any key to continue.

++

++

Relational databases hold information in tables of rows and columns. The information is retrieved by printing a specific table.

Your answer? (True or False) -> **false**

Correct!

Next Question? (y/n) **y**

++

++

A SEMANTIC TRANSITION-TREE GRAMMAR uses English sentences to retrieve database data. Semantic transition-trees are different from syntactic transition-nets in several ways.

Press any key to continue.

++

++

A semantic transition-tree grammar
a. Has some link transitions with specific words.
b. Has some link transitions specifying semantic phrases.
c. Has nodes with only one input.
d. All of the above.

Your answer? -> **c**

Your answer is not correct.

Please try again -> **b**

Your answer is still not correct.

The correct answer is : **d**

Press any key to continue.

++

++

Next Question? (y/n) **y**

++

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                    Congratulations!

                 You have finished Chapter 29 !

               Do you want to continue? (y / n)  no

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                        All Done For Now

                             Good-bye!

                        Press Any Key To End
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

FUTURE WORK

This tutorial performs well as it is written. It could, however, be improved upon to enhance its features and/or sculpt it to the needs of individual instructors. Such changes could be made, with care, while maintaining the system's characteristic simplistic appearance.

A grading system could easily be incorporated in the question evaluation portion of the system. A library of grading techniques, added to the system, would allow an instructor to choose a grading scheme reflecting his or her teaching style. The instructor could assign a numeric value to each question and extra-help question and include it as a third designation preceding the question in the question files. The grading systems could record a student's progress, questions missed and current score in the student's personal file.

The tutorial system frequently offers the option of quitting a tutoring session. A method of recording the exit point, possibly in the student's personal file, would allow the student to return to the point he or she left off. This, of course, would eliminate the frustration of repeating work already completed.

The ability to utilize messages and questions at different learning levels would greatly extend the program's flexibility. A library of questions and messages of varying difficultly or emphasis could be maintained. As the student demonstrates his or her proficiency or the lack of proficiency, questions of different files could be swapped in to adjust the emphasis. The third number rating system, mentioned above, could be used to decide when to swap and for which file.

Along this same line of thinking, the library of question files and message files could be built up by adding files over time. An instructor could pick and choose messages and questions from many different files. A method for guiding the tutorial system to each chosen message and

question would reduce the teacher's work by eliminating the need to create new question and message files for each student or class.

The repetition of the system's responses to the user eventually becomes boring, if not annoying. A library of system responses could be added along with a method of randomly choosing a response. This would alleviate the boredom. As an example, one file could contain a variety of responses for a correctly answered question. A random search method would vary the responses each time a question was answered correctly. This variety would increase the student's progress and attention span.

Many of the improvements mentioned in this section could be added as separate components. The components probably would be added gradually over time according to their need. The concept of system components is being used at the University of Massachusetts by Beverly Woolf and her colleagues(Woolf, 1991). This system might be further designed to include switches for each component so that it could be toggled "on or off" as the teacher's needs change.

Additional methods of presentation would definitely improve the system. For instance being able to include graphics and images would increase the student's understanding of the subject matter. Also needed is the ability to provide text variations such as bolding, italics, highlighting and font variety.

CONCLUSION

This thesis presents an interactive intelligent tutorial system based on a dynamic support concept. Its main objective is to provide instructors and students a pedagogical support tool for any specific college-level course. It may also be utilized by instructors of other education levels and subjects.

This tutorial system presents material organized in groupings, by chapter or subject, in the form of short messages and questions. The material is read from external files and displayed in bordered screens. Questions pertaining to the messages are displayed in sequence and are answered by the participating students. The answers are immediately evaluated with correct answers congratulated. Incorrect answers invoke the usage of the extra-help files of messages and questions.

A complete set of message, question and answer and extra-help related files, was constructed covering the fifteen major chapters of the textbook used in teaching the Introductory Artificial Intelligence course. Each message and question was meticulously composed to enhance understanding and avoid ambiguity.

Through this project, I've gained a great amount of appreciation and respect for what it takes to design an intelligent tutorial system (ITS). The importance of such systems increases with each passing day. Today's fast-paced and rapidly changing lifestyles demand quick and thorough learning, just to keep pace. ITSs will help the student keep up.

Intelligent tutorial systems will aid the already overtaxed teachers of our school systems by providing a means of extending their valuable teaching influence and their limited time. The student will be able to learn ideas and concepts outside of the classroom and at a time best suited for their own individual learning. The time to learn can be left up to the student. Teachers will also be able to add to their classroom teaching by

using tutorial systems to present concepts not covered in class due to time limitation.

Students who, for a variety of reasons, can't attend a traditional classroom environment or have trouble learning in such a situation, will also benefit from ITSs. ITS remote teaching will be available, through the computer networks, to provide the needed education in a wide variety of locations such as homes and hospitals.

ITSs can also provide the patient instruction needed by the learning-disabled student. Computer-aided tutorial systems don't have a "bad" day. They are always patient and are never frustrated by a student's inability to learn. The tutorial systems always have time to spend with the student. All of these features are very important to learning-impaired students.

Intelligent tutorial systems will greatly help teachers and students of all kinds and ages. There is a terrific future ahead for the development of educational systems.

BIBLIOGRAPHY

Bonar, J. G. (1991). Interface architectures for intelligent tutoring systems. In H. Burns, J. W. Parlett, C. Luckhardt Redfield (Eds.), *Intelligent Tutoring Systems, Evolutions in Designs*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Borland C++. Borland International Inc. (1991). Scotts Valley, California.

Burns, H. and Parlett, J. W. (1991). The evolution of intelligent tutoring systems: dimensions of design. In H. Burns, J. W. Parlett, C. Luckhardt Redfield (Eds.), *Intelligent Tutoring Systems, Evolutions in Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Chabay, R. W. and Sherwood, B. A. (1992). A practical guide for the creation of educational software. In J. H. Larkin, R. W. Chabay (Eds.), *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complimentary Approaches*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Corbett, A.T. and Anderson, J. R. (1992). LISP intelligent tutoring system: research in skill acquisition. In J. H. Larkin, R. W. Chabay (Eds.), *Computer-Assisted Instruction and Intelligent Tutoring Systems, Shared Goals and Complementary Approaches*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Cumming, G. and Self, J. (1989). Collaborative intelligent educational systems. In D. Bierman, J. Breuker, J. Sandberg (Eds.), *Artificial Intelligence and Education*. Springfield, Virginia: IOS.

Kearsley, G. (1988). Authoring systems for intelligent tutoring systems on personal computers. In D. H. Jonassen (Ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Lawler, R. and Yazdani, M. (1987). Introduction. In R. Lawler, M. Yazdari (Eds.), *Artificial Intelligence and Education, Volume One: Learning Environments and Tutoring Systems*. Norwood, New Jersey: Ablex Publishing.

Nathan, M. J., Johl, P., Kintsch, W. and Lewis, C. (1989). An unintelligent tutoring system for solving word algebra problems. In D. Bierman, J. Breuker, J. Sandberg (Eds.), *Artificial Intelligence and Education*. Springfield, Virginia: IOS.

Nelson, T. H. (1993). The right way to think about software design. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design*. Reading, Massachuetts: Addison-Wesley Publishing Company, Inc.

Ohlsson, S. (1987). Some principles of intelligent tutoring. In R. W. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education, Volume One: Learning Environments and Tutoring Systems*. Norwood, New Jersey: Ablex Publishing.

O'Malley, C. (1990). Interface issues for guided discovery learning environments. In M. Elsom-Cook (Eds.), *Guided Discovery Tutoring, A Framework for ICAI Research*. London: Paul Chapman Publishing.

O'Neil, H. F. Jr, Slawson, D. A. and Baker, E. L. (1991). Design of a domain-independent problem-solving instructional strategy for intelligent computer-assisted instruction. In H. Burns, J. W. Parlett, C. Luckhardt Redfield (Eds.), *Intelligent Tutoring Systems, Evolutions in Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Price, R. V. (1991). *Computer-Aided Instruction: A Guide For Authors*. Pacific Grove, California: Brooks/Cole Publishing Company.

Riel, M. M., Levin, J. A. and Miller-Souviney, B. (1987). Learning with interactive media: dynamic support for students and teachers. In Robert W. Lawler and Masoud Yazdani (Eds.), *Artificial Intelligence and Education, Volume One: Learning Environments and Tutoring Systems*. Norwood, New Jersey: Ablex Publishing.

Sack, W. and Soloway, E. (1991). From PROUST to CHIRON: ITS design as iterative engineering; intermediate results are important!. In H. Burns, J. W. Parlett, C. Luckhardt Redfield (Eds.), *Intelligent Tutoring Systems, Evolutions in Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Salomon, G. (1993). New uses for color. In B.Laurel (Ed.), *The Art of Human-Computer Interface Design*. Reading, Massachuetts: Addison-Wesley Publishing Company, Inc.

Schank, R. C. and Edelson, D. J. (1989). Discovery systems. In D. Bierman, J. Breuker, J. Sandberg (Eds.), *Artificial Intelligence and Education*. Springfield, Virginia: IOS.

Wager, W.and Gagné, R. M. (1988). Designing computer-aided instruction. In D. H. Jonassen (Ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Winston, P. H. (1992). *Artificial Intelligence, Third Edition*. Reading, Massachusetts: Addison-Wesley Publishing Company.

Woolf, B. (1991). Representing, acquiring, and reasoning about tutoring knowledge. In H. Burns, J. W. Parlett, C. Luckhardt Redfield (Eds.), *Intelligent Tutoring Systems, Evolutions in Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Yazdani, M. (1987). Intelligent tutoring systems: an overview. In R. Lawler, M. Yazdari (Eds.), *Artificial Intelligence and Education, Volume One: Learning Environments and Tutoring Systems*. Norwood, New Jersey: Ablex Publishing.

APPENDIX :

AN INTRODUCTION TO ARTIFICIAL INTELLIGENCE

TUTORING SYSTEM FILES

The text for each file appears here as it is structured within its file. The file's name accompanies the text in bold type and underlined.

PART I

File part1.int

```
-PART I
-Representations and Methods

     Part I introduces you to Artificial Intelligence
as a whole.  You will learn about the representations
and methods used to explain various types of intelli-
gence.  Part I teaches you about PROBLEM-SOLVING METHODS,
SEARCH METHODS, RULES and RULE CHAINING, CONSTRAINTS,
PROPAGATION and more.
****
```

Chapter 1

File p1c1.int

```
-Chapter 1
-The Intelligent Computer

     In Chapter 1 you are introduced to Artificial
Intelligence as a whole.  The definition of Artificial
Intelligence is explored to give you an idea of its
importance as a branch  of engineering as well as a type
of science.  You will learn about projects successfully
using Artificial Intelligence.  You will also learn the
criteria to judge the success of an application.
****
```

File p1c1.q

```
^1
1 t  Artificial Intelligence is concerned with pursuing
     the concept that computers can behave in a manner
     that humans perceive as intelligent.
%1
^2
2 m  Some uses for Artificially Intelligent computers are:
     a.  To more economically and safely design and test
           new devices .
     b.  To solve difficult problems more quickly and effic-
           iently than humans can.
     c.  To allow space exploration without the risk to
           human life.
     d.  All of the above.
%2
3 m  Even now intelligent computer systems are able to:
     a. Design new devices
     b. Solve analysis problems
     c. Learn from examples
     d. All of the above.
     e. Answers a and b.
%3
```

```
^3
4 m  When a person decides how to act in a specific
     situation, he or she may REMEMBER a previous
     similar situation, the action taken then and the
     action's consequences.  A computer may be
     programmed to do the same by:
     a.  Asking the user to remember for it.
     b.  Using the Internet and asking another computer.
     c.  Searching library files for a similar situation,
           then applying the rules and stored actions to the
           new situation.
     d.  None of the above.
%4
****
```

File p1c1.a

```
1 true
    For computers to be valuable helpers or to reason
  on their own, they need to behave and reason as a
  human would.
2 d
3 d
4 c
****
```

File p1c1.m

```
1     One definition of Artificial Intelligence as
   stated by Feigenbaum and Mc Corduck is:

      "A subfield of computer science that concerns
   pursuing the possibility that the computer
   can be made to behave in the ways that humans
   recognize as "intelligent" behavior in each
   other."
++++
2     There are many reasons why Artificial Intelligence
   should be used in programming computers.  One reason
   is to perform tasks dangerous to humans.  For instance,
   a computer can test the heat resistant properties of
   a type of plastic to be used on instruments to be
   sent into a volcano.  The alternative might be to
   send the scientist and a sample of the plastic.
++++
3     When a person encounters a problem or situation
   requiring decision making, he or she draws on current
   facts, impressions, and the results of previous actions
   and decisions to help make that decision.  Artificial
   Intelligence programming is the way computers gain
   and use these same facts and results of previous
   actions and decisions.
****
```

File p1c1.exm

```
1     Humans want to program computers to make decisions
   and act on those decisions as people do.  Artificial
   Intelligence, a part of computer science, is concerned
   with learning how people learn and make decisions
   with the goal of programming computers to simulate
   human thinking.
++++
2     Computers using Artificial Intelligence can help
   people by performing tedious, boring and dangerous
   tasks.  Computers can analyze data in efficient,
   patient and unbiased ways.  Computers also do not

   require rest breaks, they can work around the
   clock and don't take vacations.
++++
3     Computer programming and Artificial Intelligence
   has come along way towards simulating human reasoning.
   The ID3 system has produced identification rules used
   in disease diagnoses and credit assessment.  The
   START system was employed to help journalists and
   scientists understand the data and the voyage of
   the space craft Voyager 2 and its encounter with
   the planet Neptune.
++++
```

```
4     Acting like a human brain, a computer can be
   instructed to store its experiences in library
   files, its equivalent of a memory.  Programs can be
   written to "decide" which previous experiences are
   similar enough to a current situation to use.  The
   computer then may use the stored action to solve
   the current problem possibility recording the
   results in the library file to be used later.
****
```

File p1c1.exq

```
^1
1 t  Artificial Intelligence is based on the idea
     that computers can be programmed to behave
     intelligently.
^2
2 m  A robot with Artificial Intelligence is a good
     assembly line worker because:
     a.  It can do repetitive and boring tasks
           efficiently without tiring.
     b.  Additional human jobs are required to
           supervise the robot.
     c.  More human jobs are created to maintain
           the robot and the required paper work.
     d.  None of the above.
^3
3 y  Is it possible for a robot using Artificial
     Intelligence to examine the symptoms of a sick
     person and make a intelligent diagnosis of the
     disease causing the symptoms?
^4
4 t  A robot can use library files as a human uses
     his or her arms and legs.
****
```

File p1c1.exa

```
1 true
2 a
3 yes
4 false
     A library file is like a human's memory.  It
  stores the computer's previous "experiences".
****
```

Chapter 2

File p1c2.int

```
-CHAPTER 2
-Semantic Nets and Description Matching

     Chapter 2 teaches you the importance of a good
REPRESENTATION.  You will learn how to gage the quality
of a representation.  Chapter 2 will give you a good
understanding of SEMANTIC NETS and the important problem-
solving method called "DESCRIBE AND MATCH".  You will
also be introduced to a GEOMETRIC-ANALOGY PROBLEM SOLVER,
a plot recognizer and an object identifier based on
feature recognition.
****
```

File p1c2.q

```
^1
1 m  A good representation must:
     a.  Clearly define the problem's objects,
          relationships and inherent constraints.
     b.  Present only relevant important details.
     c.  Presents the problem simply, concisely and
           completely.
     d.  All of the above.
%1
^2
2 m  The node-and-link representation is good to use
     for the farmer, fox and goose problem because it:
     a.  Explicitly describes the important objects
```

```
        and their relations.
     b. Clearly presents the problem's inherent
        constraints.
     c. Is easy to understand and easy to find the
        problem's solution.
     d. All of the above.
%2
^3
3 t  Lexically a semantic net has constructor procedures
     to create the objects and links used.
%3
4 t  A diagram conveniently shows the structural part
     of a semantic net representation by depicting the
     objects as circles and the links as arrows.
%4
^4
5 m  To identify an object, the feature-based object
     identification method :
     a. Compares the characteristics of the object with
        those in the description library.
     b. Determines the smallest distance between the
        unknown object and some identified object in the
        feature space.
     c. Gives each feature a numerical value, sums the
        feature values then compares the sum with the
        values in the description library.
     d. None of the above.
%5
^5
6 t  One method for handling an analogy problem, using
     describe-and-match, starts by describing rules on how
     one object becomes another.
%6
****
```

File p1c2.a

```
1 d
2 d
3 false
4 true
5 c
6 true
****
```

File p1c2.m

```
1     A REPRESENTATION is a set of conventions which
   describe a class of things.
++++
2     A DESCRIPTION uses the conventions of a represen-
   tation to describe a specific thing precisely and
   clearly.
++++
3     A REPRESENTATION has four main parts:
   - The LEXICAL part describes the vocabulary of
     the representation, the symbols used.
   - The STRUCTURAL part tells the way the symbols
     may be presented and the constraints on the
     arrangement of the symbols.
   - The PROCEDURAL part has the procedures to allow
     manipulation of the descriptions, their creation,
     modification and answering questions using them.
   - The SEMANTIC part creates a method of associating
     the descriptions and meaning.
++++
4     The DESCRIBE-AND-MATCH METHOD describes an object
   using an appropriate representation.  A description
   library is then searched for a satisfactory matching
   description.  The result of the search is announced,
   match found or failure.
++++
5     A DESCRIBE-AND-MATCH ANALOGY REPRESENTATION
   creates rules describing how A is transformed into B.
   The rules of how C is transformed into the answer are
   then compared with the rules of the A to B transfor-
   mation.  The best answer occurs with the best matching
   of rules.
****
```

File p1c2.exm

```
1     A CONVENTION is a "customary practice, rule or
  method.  A REPRESENTATION uses a set of rules and
  methods to describe a general group of objects.
  For example, a representation might describe "cars"
  as a group of objects.
++++
2     A DESCRIPTION uses the sets of rules and methods
  of a REPRESENTATION  to describe a particular object
  within the representation's group of objects.  For
  instance, a description might present a Lamborgine
  as a particular object in the group in the represen-
  tation of "cars".
++++
3     The LEXICAL part of a semantic net consists of:
  - the nodes which symbolize the objects,
  - the links which show the relationships between the
    objects, and
  - the link labels which denote particular relation-
    ships or constraints applied to the relationship
    between the objects.
++++
4     The STRUCTURAL part of a semantic net is the
  pieces of the diagrams showing the net.  The nodes
  are often shown as rectangles, ellipses, or circles.
  Links are shown as lines between the nodes, with
  arrowheads at one or both ends.
++++
5     The DESCRIBE-AND-MATCH METHOD uses a set of
  library descriptions.  The description of an
  unknown object is compared to each of the object
  descriptions in the library.  If a library descrip-
  tion matches the unknown object's description, the
  object is identified otherwise failure is announced.
++++
6    FEATURE-BASED OBJECT IDENTIFICATION, is a type of
  DESCRIBE-AND-MATCH which uses object features and
  places them in a feature space.
****
```

File p1c2.exq

```
^1
1 t A class of objects is described in a representation by
    a set of methods and rules (a set of conventions).
^2
2 m A good description:
    a. Clearly defines a particular object within a group.
    b. Contains no irrelevant or redundant characteristics.
    c. Must describe a particular group of objects.
    d. Answers a and b.
^3
3 t The lexical portion of a semantic net contains the
    symbols of a representation, the nodes, links, and
    link labels.
^4
4 t Lines with arrows from one object to another
    illustrates the way symbols may be arranged and is
    part of the structural part of a semantic net
    representation.
^5
5 m To identify the make of a car he has seen, Jack writes
    down a description of the car. He goes to the library
    and chooses a book on cars. He leafs through the pages
    until he comes to a picture and the statistics of the
    car he has seen. Jack is using:
    a. Means-Ends Analysis
    b. In-Depth Search
    c. Describe-and-Match
    d. Brute-Force Search
^6
6 t  An analogy problem using describe-and-match compares
     the rules for transforming an object X into object Y
     with the rules for transforming an object T into
     object S.
****
```

File p1c2.exa

```
1 true
2 d
```

```
3 true
4 true
5 c
6 true
****
```

Chapter 3

File p1c3.int

```
-CHAPTER 3
-Generate and Test, Means-Ends Analysis
-and Problem Reduction

     Chapter 3 introduces you to three problem-solving
methods:  GENERATE-AND-TEST, MEANS-ENDS ANALYSIS, and
PROBLEM REDUCTION.  Two new representations, the STATE-
SPACE REPRESENTATION and the GOAL TREE REPRESENTATION,
are introduced by learning about problem reduction.
****
```

File p1c3.q

```
^1
1 t The tester must always wait until the generator
    has completed compiling all possible solutions

    before it can begin testing the proposed
    solutions.
%1
2 t A generate-and-test program may stop when a
    certain number of acceptable solutions are
    reached or after the first acceptable solution
    is found.
%2
3 m When the generate-and-test method is used in
    identification -
    a.  The generator produces hypotheses to be
          tested.
    b.  The generator is not used.
    c.  It is not a very efficient choice of methods.
    d.  None of the above.
%3
^2
4 m A good generator is:
    a.  Concise, restrictive, and informative.
    b.  Complete, non-redundant, and informed.
    c.  Patient, forgiving, and understanding.
    d.  None of the above.
%4
5 y A good generator is complete in that it will
    eventually produce all possible solutions.
%5
6 t A good generator is sometimes redundant to
    emphasize the important aspects of a problem.
%6
7 y A good generator is informed when it is able to
    restrict the possible solutions by using a set of
    limiting information.
%7
****
```

File p1c3.a

```
1 false
2 true
3 a
4 b
5 yes
6 false
7 yes
****
```

File p1c3.m

```
1       The GENERATE-AND-TEST method has two main parts.
   The GENERATOR lists all of the possible solutions.
   The TESTER then checks each proposed solution with
```

the possible solutions to either accept or reject it.
++++
2 The efficiency of a generate-and-test program depends greatly on its generator. A good generator must be complete, non-redundant and informed.
****
3 The MEANS-ENDS ANALYSIS method identifies a procedure to get from one state to the goal or from one state to the next intermediate state in the path to the goal.
++++
4 PROBLEM-REDUCTION and GOAL REDUCTION are two names for a method of achieving a desired goal. Problem-reduction makes reaching the desired goal easier by reducing the goal to smaller simpler subgoals then solving each subgoal.
++++
5 The representation used in problem-reduction and goal reduction is the GOAL TREE. The goal tree is a semantic tree with parent, child, root and leaf nodes. CONSTRUCTORS connect the nodes with branches. READERS list a node's children or a node's parent. The nodes of a goal tree are the goal and subgoals while the links or branches are the methods to achieve each goal.
++++
6 An AND-OR TREE has subtrees in which all child nodes must be satisfied to satisfy the parent node's goal. These are AND subtrees. Other subtrees are OR subtrees in which any of the child nodes may satisfy the parent node's goal.
****

File p1c3.exm

1 The generator may be required to generate all possible solutions before the tester is allowed to start. More often a possible solution is generated then tested before another possible solution is generated.
++++
2 In many situations it is unnecessary and inefficient to generate all possible solutions to a problem. Usually generation of possible solutions stops when the first acceptable solution is found.
++++
3 Identification of objects, i.e. trees, phone numbers, names in a phone book, is very useful in saving the time spent in searching the appropriate location. In identification, the generator produces hypotheses, theories, to be tested. The tester then tests the hypotheses for a solution.
++++
4 A good generator is COMPLETE when it will eventually provide all of the possible solutions to a given problem. It is NON-REDUNDANT because it will never produce the same solution more than once. And, a good generator is INFORMED when it is able to restrict the solutions it presents by using possibility-limiting information.
++++
5 COMPLETENESS, in a generator, means that it can produce ALL possible solutions to a problem. Usually this is not needed, but if a solution is not produced that may be the needed (acceptable) even crucial solution.
++++
6 A good generator is never redundant. A computer won't "miss" a solution because it "blinked" or was distracted as a human might be. Hence, redundancy is always a waste of computer time which could be critical if a solution must be found quickly. A good generator is NON-REDUNDANT.
++++
7 To be informed, a generator is able to use stored information to limit the number of possible solutions it must generate. For example, a generator, trying to identify a wild flower, will not generate solutions involving red flowers when the flower it is seeking is blue.
****

File p1c3.exq

```
^1
1 m  Which of these examples best illustrates a generate-
     and-test paradigm.
       a.  To find a lost earring you look under pillows,
           under the bed, behind the cookie jar.
       b.  To identify a flower, you leaf through a book of
           flowers looking at each picture until one matches
           the flower.
       c.  Neither a or b illustrates a generate-and-test
           paradigm.
       d.  Both a and b.
^2
2 y  For a particular problem, the generator of a generate-
     and-test problem solving program, can potentially
     generate 200 possible solutions.  150 of these sol-
     utions are acceptable but any acceptable solution is
     satisfactory.  Is it necessary to generate all the
     possible solutions in this case?
^3
3 2  An example of the use of a generate-and test paradigm
     in identification is:  Looking for the name of a wild
     flower in a wild flower book.
^4
4 t  To be a good generator, it must be able to selectively
     limit the solutions it produces, never present the
     same solution more than once, and be able to generate
     all possible solutions if necessary.
^5
5 y  A generate-and-test program is used in a airport to
     tell pilots if they may land, on which runway and if
     they can't land, which direction to fly.  To do this
     the generator is able to produce all of the possible
     combinations of all situations each time a plane
     appears on radar.  Is this generator complete?
^6
6 m  Redundancy in a generator is never desirable because:
       a.  It's hard to program.
       b.  It can't be controlled.
       c.  It wastes time, is inefficient, and unnecessary.
       d.  All of the above.
^7
7 y  A particular generator provided Joe, who was interested
     in buying a house in Chicago, with the addresses of
     all houses for sale in the city even though he wanted
     to live by the lake.  Is this generator informed?
****
```

File p1c3.exa

```
1 b
2 no
3 true
4 true
5 yes
6 c
7 no
****
```

Chapter 4

File p1c4.int

```
-Chapter 4
-Nets and Basic Search

     In Chapter 4, you will learn to solve search
problems by finding paths through nets.  Two groups
of methods are presented.  The BLIND SEARCH methods
include DEPTH-FIRST SEARCH and BREADTH-FIRST SEARCH.
     You will also learn to limit searches using
HEURISTIC SEARCH methods.  HILL-CLIMBING, BEAM SEARCH
and BEST-FIRST SEARCH use heuristic estimates to
intelligently limit a search.
####
     By the end of Chapter 4 you will know:
  - If searching is the best solution for the problem.
```

```
   - The search method to solve the problem.
   - The most efficient search method to solve the problem.
****
```

File pic4.q

```
^1
1 m A representation using nodes to denote paths and
    branches or links to show path extensions, is
    called a:
    a.  Search Map
    b.  Flow Diagram
    c.  Search Tree
    d.  Path Search
%1
2 t A partial path in a search tree is a path which
    goes to a node with no branches leading away from
    it.
%2
^2
3 t Depth-first search intelligently considers all path
    alternatives at each node before choosing the next
    path extension.
%3
4 m With a depth-first search, if the goal is not reached
    by the time a leaf node is reached:
    a.  The search is ended with no path found.
    b.  The search returns to the leaf's nearest ancestor
          with an unvisited child node.
    c.  The search returns to the root node to
          try another child of the root.
    d.  Answers b and c.
%3
^3
5 m Breadth-first search:
    a.  Considers all nodes on a single level before
          descending to the next level.
    b.  Visits all of a node's children before extending
          the path one step more.
    c.  Considers all of a node's descendants before
          its sibling nodes.
    d.  Answers a and b.
%4
^4
6 m To plan a trip from Los Angeles to New York you use
    a search tree listing all possible paths between
    the two cities.  Which search should you use to find
    a path?
    a.  Nondeterministic search
    b.  Depth-first search
    c.  Breadth-first search
    d.  None of the above.
%5
^5
7 t Heuristic search methods can shorten a search by
    choosing path extensions most likely to reach the
    goal.
%6
^6
8 m Which type of search uses the remaining distance
    to the goal in deciding which child node to visit?
    a.  Hill-Climbing
    b.  Depth-First
    c.  Best-First
    c.  Beam Search
%7
^7
9 m Beam search:
    a.  Considers all the children of a node choosing
          the best.
    b.  Chooses a node randomly in a level to expand
          the path.
    c.  Chooses n best nodes of a level to expand.
    d.  None of the above.
%8
^8
10 t Best-first search considers all possibilities it
     has seen so far in the search when deciding which
     node to expand.
****
```

File p1c4.a

```
1 c
2 false
3 false
4 b
5 a
6 a
7 true
8 a
9 c
10 true
****
```

File p1c4.m

```
1     Many problems, such as finding a path through
   several cities, may be represented by special
   semantic trees called SEARCH TREES.  To solve the
   problem a path must be found from the root node
   to the goal node.
++++
2     DEPTH-FIRST SEARCH assumes that all paths are
   equal.  It starts at the root and blindly chooses
   a child node for each parent node to move down
   though the tree on its journey to the goal.  Depth-
   first search is a BLIND SEARCH because there are
   no rules (heuristic) used to choose the next node
   to move to.
++++
3     BREADTH-FIRST SEARCH, another BLIND SEARCH,
   moves across a search tree visiting a node's siblings
   before its children.  An entire level is searched
   before moving on to the next deeper level in the tree.
++++
4     A NONDETERMINISTIC SEARCH may be the best search
   if you have no clue about the tree structure or the
   path needed.  Nondeterministic search is actually a
   random search.  Each time the path is extended, it
   is extend to a neighbor node, which is randomly chosen.
++++
5     HEURISTIC SEARCH methods use rules or conditions
   to make the decisions to extend a path.  The search
   efficiency is improved because the most plausible
   paths are explored first, essentially eliminating
   the least-likely and useless paths.
++++
6     HILL-CLIMBING, a heuristic search, is a special,
   informed, version of DEPTH-FIRST SEARCH.  At each
   point that a child node is chosen, the choices are
   ordered.  A heuristic measure of the remaining
   distance to the goal is applied to choose the short-
   est path.
++++
7     BEAM SEARCH, also a heuristic search, is similar
   to breadth-first search.  It searches the tree level-
   by-level.  The number of nodes is reduced, however,
   by moving down through only the best nodes on each
   level.  All other nodes are ignored.
++++
9     Another heuristic search method is BEST-FIRST
   -SEARCH.  In its search, Best-first chooses the best
   node "so-far" by using a heuristic rule.  The node
   chosen need not be a neighbor but may be located
   anywhere in the already "partially developed" tree.
****
```

File p1c4.exm

```
1    SEARCH TREES are often used to calculate routes
  and distances from one place, the source, to another
  place, the goal.  Each node represents a place where
  a decision must be made.  For example: Is it shorter
  to turn right, left or continue straight ahead.  The
  links between modes designate the possible routes to
  the next node(s).  These usually have a value assoc-
  iated with them. (The distance to the next town, for
  instance.)
++++
2     SEARCH TREES are special semantic trees.  They
  begin with a ROOT NODE which has no PARENT NODE.
  LEAF NODES have no child nodes.  Movements from one
```

```
   node to another along connecting branches are PATHS.
   A COMPLETE PATH is a path from the root node to the
   GOAL NODE.  A PARTIAL PATH is a path does not reach
   the goal.
++++
3    DEPTH-FIRST SEARCH blindly travels down a tree
   from the root to the leaves.  If the goal is not
   reached, it returns to the nearest ancestor with an
   unvisited child node.  Again it descends the tree to
   the leaves.  This is repeated until the goal is
   reached or until all of the nodes are visited.  The
   path is extended randomly, blindly, at each node.
++++
4    BREADTH-FIRST SEARCH visits all nodes on a par-
   ticular level before it randomly extends the path
   one step to the next level.
++++
5    A NONDETERMINISTIC SEARCH randomly chooses a node
   to expand.  It keeps the search from exploring too
   many branches or too many levels.  It is a middle of
   the road search between depth-first and breadth-first
   searches.
++++
6     A HEURISTIC is a "rule-of-thumb" or a condition
   used to make intelligent decisions.  For example, at
   an unfamiliar intersection, you probably would choose
   the road leading north since you know your destination
   is north.
++++
7    An example of a HILL-CLIMBING search is if you were
   at an intersection with distance signs for each road.
   Left is 5 miles to a town in the direction of your
   destination.  Right is 7 miles to another town also
   toward your destination and ahead is a town 2 miles.
   You know your destination is 10 miles away.  Your
   choice should be right since there is only 3 miles
   from the town to your destination.
++++
8    BEAM SEARCH considers each node on a level then
   picks a number of the best nodes to move down to the
   next level.  All other nodes are ignored.  This shortens
   the search to only the best choices to be considered.
++++
9    A queue is often used in a BEST-FIRST SEARCH to
   store the nodes already visited.  The best node, no
   matter where it is located, is expanded.  This will
   likely find shorter paths than other search methods.
****
```

File pic4.exq

```
^1
1 y A search tree is often used to arrange all poss-
    ible paths from one place, the start node, to
    another, the goal node.
^2
2 m Jane used a search tree to discover all the routes
    from Omaha to Lincoln.  She found one path which
    began in Omaha and ended in Sioux City, Iowa.
    Is this path a
    a.  Branching Path  or
    b.  Partial Path    or
    c.  Scenic Path     or
    d.  Complete Path
^3
3 t The depth-first search is called a blind search
    because, at each node, a child node is randomly
    chosen to visit.
^4
4 t Depth-first search will visit the root node's
    second child before it visits any leaf node.
^5
5 y All paths of a certain length are searched by the
    breadth-first search before a longer path is
    considered.
^6
6 t A nondeterministic search is best used when the tree
    structure is unknown, may have a large branching
    factor or may have a great depth.
^7
7 m A heuristic is:
    a.  The direction a path takes from the root to
          the goal.
```

```
    b.  A rule or condition which a decision is
          based on.
    c.  Neither a or b.
    d.  Both a and b.
^8
8 y An intelligent version of the depth-first search
    is the hill-climbing search since it uses a
    heuristic to make its decisions.
^9
9 t All of the nodes in a tree level are eventually
    explored with the best choices explored first when
    beam search is employed.
^10
10 y In a best-first search, must the next path
     extended be the path just extended.
****
```

File p1c4.exa

```
1 yes
2 b
3 true
4 false
     Depth-first travels down a tree before it travels
  across.  Thus, visiting a leaf node before the root's
  second child node.
5 yes
6 true
7 b
8 yes
9 false
10 no
****
```

Chapter 5

File p1c5.int

```
-CHAPTER 5
-Nets and Optimal Search

   Chapter 5 introduces you to OPTIMAL SEARCHING and
four more searches:  BRITISH MUSEUM, BRANCH-AND-BOUND,
DISCRETE DYNAMIC PROGRAMMING and A*.  These procedures
are used when the cost of traveling a path is very
important.  The best paths, not just any path, are
found by these searches.
****
```

File p1c5.q

```
^1
1 t  British Museum is a very efficient procedure
     because it finds every possible path.
%1
^2
2 m  In a branch-and-bound search, the first path
     reaching the goal is shortest because:
     a.  All partial paths so far are considered at
           each path extension.
     b.  The shortest partial path is chosen to be
           extended each time.
     c.  All newly extended partial paths are considered
           along with previously extended paths.
     d.  All of the above.
%2
^3
3 t  When using underestimates in a branch-and-bound
     search, the smallest estimates are always extended.
%3
4 t  A search using branch-and-bound is ended when all
     partial path estimates are greater than the lowest
     complete path distance so far encountered.
%4
^4
5 m  Branch-and-bound is improved when dynamic program-
     ming is added because:
     a.  Only the shortest branch of a partial path's
           end node is extended.
```

```
     b.  An entire level may be ignored if the branches
           of that level are too long.
     c.  A search ends when a branch to be extended is
           longer than an estimate.
     d.  All of the above.
%5
^5
6 y  Dynamic Programming can eliminate identical paths
     by eliminating all paths coming into a node except
     the path with the least cost.
%6
^6
7 t  Dynamic programming improves the efficiency of A*
     by limiting the number of branches to extend.
%7
8 m  A* uses a lower-bound estimate of:
     a.  The distance remaining to the goal.
     b.  The number of remaining nodes to be visited to
           the goal.
     c.  The total distance from the root to the goal.
     d.  The remaining distance minus the distance
           already traveled.
%8
****
```

File p1c5.a

```
1 false
    The British Museum will find all possible paths
  and pick the best but it is very inefficient because
  there are no heuristics to limit the search.
2 d
3 true
4 true
5 a
6 yes
7 true
8 c
****
```

File p1c5.m

```
1     The BRITISH MUSEUM SEARCH is a brute-force
   method to find the best path to the goal.  It may use
   depth-first or breadth-first search modified to find
   all possible paths.
++++
2     The BRANCH-AND-BOUND optimal search looks for
  the best path by keeping track of all partial paths,
  thus far.  The shortest path is extended one level
  through all of its branches.  These new partial paths
  are now added to and considered with the old partial
  paths to choose the next shortest.  The search
  continues until a shortest partial path is longer
  than the shortest complete path.
++++
3     Adding UNDERESTIMATES can sometimes improve branch-
  and-bound.  The estimated total path length is equal to
  the distance already traveled plus an estimate of the
  distance left to travel to the goal.

  e(total path length) = d(already traveled) +
                                 e(distance remaining).
++++
4     The DYNAMIC PROGRAMMING PRINCIPLE states that on
  your way from the root to the goal, you may ignore all
  paths from the root to any intermediate node if that
  path is not the shortest path to that intermediate node.
  This principle also removes redundant paths improving
  efficiency.
++++
5     To remove redundant paths, DYNAMIC PROGRAMMING says,
  "If two or more paths reach a common node, delete all
  those paths except the one that reaches the common
  node with the minimum cost."
++++
6     The procedure A* combines BRANCH-AND-BOUND using
  a REMAINING DISTANCE ESTIMATE with DYNAMIC-PROGRAMMING.
  A lower-bound is usually used to estimate the remaining
  distance.
****
```

File p1c5.exm

```
1     While the BRITISH MUSEUM SEARCH does find all
   possible paths to the goal, this property is seldom
   necessary.  Searching a large tree using British
   Museum means checking every possibility.  A time
   consuming and inefficient prospect.  Using a
   heuristic to limit the search is a much better
   choice.
++++
2     BRANCH-AND-BOUND assumes you know something about
   the problem, for instance, the distance and a route
   from Omaha to Des Moines.  All possible paths between
   the two cities are represented in a tree.  (All
   partial paths are kept track of for possible extent-
   ion.)  As the search moves from the root (Omaha), the
   shortest partial path is extended at each node (city)
   adding as many new partial paths as possible until the
   search reaches the goal (Des Moines).  If a partial
   path exceeds the value of the known complete path
   it is dropped.  Because only the shortest partial
   paths are extended, the path reaching the goal first
   is likely to be the shortest.
++++
3     An estimate of the distance remaining to the goal
   can be incorporated into the BRANCH-AND-BOUND SEARCH.

   e(total path length) = d(already traveled) +
                          e(distance remaining).

   If the estimate is close, using underestimates can
   improve the search by limiting the paths to extend
   to the shortest partial paths.
++++
4     It makes no sense extending a path beyond the
   length of a known path reaching the goal.  It is
   then no longer the shortest path.
++++
5     DYNAMIC PROGRAMMING serves to augment branch-and-
   bound by limiting the extension of the already chosen
   shortest path to the shortest next extension.  A
   path with a possible 5 path extensions is limited to
   a single extension, the shortest.
++++
6     More than one path between two nodes wastes time
   and efficiency.  For example, if four routes con-
   nected Omaha to  Lincoln, and you were interested
   in getting between the two cities in the least
   amount of distance you would naturally choose the
   shortest of the four and  ignore the rest.  This
   is what DYNAMIC PROGRAMMING does by eliminating all
   but the least cost paths ending at a common node.
++++
7     A* uses a queue to store all partial paths.  The
   paths are sorted by length with least cost at the
   front of the queue.  The branch-and-bound portion of
   A* removes the end node and extends the path to all
   of its neighbors.  Dynamic programming then rejects
   paths with loops and deletes any paths ending at the
   same node except the one with the least cost.  The
   remaining paths are added to the queue.  The queue
   is resorted.
++++
8      The LOWER-BOUND ESTIMATE is combined with the
   branch-and-bound portion of A*.  The estimate is
   used in sorting the queue.  The estimate is added
   to each path length then the queue is sorted.
****
```

File p1c5.exq

```
^1
1 t  The British Museum is a cumbersome technique and
     should be used only on small search trees.
^2
2 m  Branch-and-bound search can be as much work as
     British Museum when:
     a.  All possible partial paths are extended to
           find the shortest path.
     b.  All possible paths are the same length.
     c.  Answers a and b.
     d.  None of the above.
^3
```

```
3 t  The optimal path is guaranteed, when using under-
     estimates because only the longest partial paths
     are extended.
^4
4 m  Two complete paths are known to a particular goal.
     Path A has a length of 12, path B's length is 9.
     At one point in a branch-and-bound search, 3 new
     partial paths are encountered.  X's length is 7,
     Y's length is 10 and Z's length is 5.  Which paths
     should be extended?
     a.  All three paths.
     b.  Path X first then Z, if necessary.
     c.  Path Z first then Path B, if necessary.
     d.  Path Z.
^5
5 2  Dynamic programming improves the efficiency of
     branch-and-bound by eliminating all but one
     branch from each partial path node.
^6
6 1  Dynamic Programming:
     a.  Removes the redundant paths and improves
           efficiency of a search.
     b.  Can be combined with the branch-and-bound
           search to limit unnecessary searching.
     c.  Both answers a and b.
     d.  Answer a only.
^7
7 2  The lower-bound estimate provides a measure to
     decide which branches to extend toward the goal.
^8
8 3  A lower-bound estimate eliminates all paths with
     costs below the estimate.
****
```

File p1c5.exa

```
1 false
2 c
3 false
4 d
5 true
6 c
7 true
8 false
****
```

Chapter 6

File p1c6.int

```
-Chapter 6
-Trees and Adversarial Search

     In Chapter 6, you will learn how programs play
games using a type of semantic tree of choices, a
GAME TREE.  You will learn the basic decision making
method,  MINIMAX SEARCH, and how to reduce the search
using the ALPHA-BETA PRUNING procedure.
****
```

File p1c6.q

```
^1
1 m The ply of a certain game tree is 11.  What does
    this mean?
    a.  The tree has a flexibility rating of 11 choices
          per level.
    b.  The number of nodes in the tree is 11.
    c.  The number of levels in the tree plus the root
          level is 11.
%1
2 y The branches of a game tree keep a running score as
    the players take turns making decisions.
%2
^2
3 t The competition of a game is seen in the minimizing
    and maximizing levels of a game tree.
%3
```

```
4 m A look ahead procedure
    a.  Examines the next level down in a game tree to
          make a decision in the current level.
    b.  Is used in a minimax search.
    c.  Allows a player to make a choice based on what
          his opponent's choices are.
    d.  All of the above.
%4
5 m The minimax procedure relies on
    a.  Giving a quality numeric value to all board
          situation judgments.
    b.  Negative quality values represent one player,
          positive values belong to the other player.
    c.  Both a and b.
    d.  Neither a nor b.
%5
^3
6 m During alpha-beta pruning, when the program is at
    a minimizing level it
    a.  Blindly chooses the least static evaluation
          to prune a tree branch.
    b.  Checks what is known about a node's ancestor
          nodes before accepting the value or pruning
          the branch.
    c.  Prunes a branch if the choice is a bad one.
    d.  Answers b and c.
%6
7 t Alpha-beta pruning follows the idea that if a move
    seems bad, it is best to check it out just to be
    sure.
%7
*****
```

File p1c6.a

```
1 c
2 no
3 true
4 d
5 c
6 d
7 false
****
```

File p1c6.m

```
1     In a game, such as chess or checkers, one decision
   leads to another resulting in a tree.  This is a
   special semantic tree called a GAME TREE.  Its NODES
   hold the board configurations, the BRANCHES show how
   a configuration is changed into another configuration
   in a single move.  The decisions are made by two
   adversaries (players) who take turns.
++++
2     Games are played by two players, one trying to
   MAXIMIZE his or her position the other trying to
   MINIMIZE his or her opponent's position.  Game
   trees are arranged with minimizing and maximizing
   levels representing each player's strategy.  The
   players make choices to further their own objectives
   but also taking into consideration the other player's

   available choices one level down in the tree.  This
   is the LOOKAHEAD procedure of the MINIMAX SEARCH.
++++
3     Because the MINIMAX is expensive, a procedure
   called ALPHA-BETA PRUNE is used.  Alpha-beta uses
   information already known about a node's ancestors
   to decide whether a particular game tree branch
   should be further explored or abandoned (PRUNED).
****
```

File p1c6.exm

```
1     A GAME TREE is a semantic tree with nodes repre-
   senting ways the game board can be arranged.  The
   branches tell how one board arrangement is changed
   into another with just one move.  The PLY (p) of
   game tree is the depth (d), the number of levels,
   plus the root level.
                    p = d + 1
```

```
++++
2     The BRANCHES of a game tree connect the nodes
   containing board configurations.  The branches in-
   dicate how one board configuration becomes another
   through one player's single decision or MOVE.
++++
3     The MAXIMIZING levels contain the choices a
   maximizing player has to increase his or her ob-
   jective when the minimizing player's choices are
   considered in the next level down in the tree.  The
   reverse is true of the MINIMIZING levels.  Compe-
   tition is the decisions made by players to max-
   imize their advantage and minimize their opponent's.
++++
4     The MINIMAX SEARCH uses a LOOKAHEAD PROCEDURE.
   A player uses the values of the next layer down (his
   opponent's choices) to make his own choices.  The
   choice values of one level effect the score in the
   next level up.
++++
5     The MINIMAX PROCEDURE uses STATIC EVALUATION to
  compute a quality number for all judgments made with
  regard to board situations.  Positive numbers show
  "favor" to one player while negative numbers show
  "favor" to the other player.  The amount of "favor"
  is the absolute value of the quality number.
++++
6     During the alpha-beta procedure, the next level up
  is checked to ensure the validity of choosing a
  certain move.  For instance in this partial tree,
  3 is chosen as maximum          =5  O     max
  in the left branch leaves          /  \
  and passed to the minimum                O   min
  level.  The top Max level             /   |  \
  knows that it can't do any    =3  O       O      O  max
  better than the 3 which is        / \    / \    / \
  less than the 5 it already       1   3  9  2 9    4
  has.  So it abandons the branch (prunes it) and
  accepts the 5.
++++
7     The whole idea behind the ALPHA-BETA PRUNE pro-
  cedure is that it wastes time, energy and efficiency
  to proceed along a game tree branch if the move
  appears to be a wrong one.  If the move seems bad,
  that branch is abandoned and the procedure moves on.
****
```

File p1c6.exq

```
^1
1 m1 A certain game tree has a ply of 15
     a.  It has a depth of 14 levels.
     b.  15 players can compete using this game tree.
     c.  It has a depth of 14 levels plus its root.
     d.  None of the above.
^2
2 m A move
     a.  Is the same as a game tree branch.
     b.  Is when one subtree is moved to another game
           tree.
     c.  Is how one board configuration is changed into
           another.
     d.  Answers a and c.
^3
3 t A game tree is organized into maximizing and min-
     imizing levels according to which player's turn
     it is.
^4
4 m In this game tree, what choice should the maximizing
     player make on the first MAX level using a look-
     ahead procedure.
     a.  2                          O          max
     b.  1                        /   |   \
     c.  3                     O      O      O  min
     d.  All choices          / \    / \    / \
         are equally        O   O  O   O  O   O
         equally good.     15   25 30  10 20  5
^5
5 y The minimax procedure uses a static evaluator to
     assign a static evaluation score to each board
     situation (or mode) in a game tree.
^6
```

```
6 m In this game tree which branch will be pruned?
    a.  A and C                  O          max
    b.  C and E                /   |   \
    c.  A and D              O     O     O  min
    d.  D and F           A/ \B  C/ \D  E/ \ F
                          O   O  O   O  O   O
                          5   7  2  10  1   3
^7
7 t The alpha-beta procedure makes the minimax search
    more efficient by eliminating tree branches con-
    taining moves which are bad.
****
```

File pic6.exa

```
1 a
2 d
3 true
4 b
5 yes
6 d
7 true
****
```

Chapter 7

File pic7.int

```
-Chapter 7
-Rules and Rule Chaining

     Chapter 7 will teach you how to solve problems
using IF-THEN RULES.  You will learn about FORWARD
CHAINING and BACKWARD CHAINING.  By the end of Chapter
7, you will know when forward chaining should be used
and when backward chaining is best used.
****
```

File pic7.q

```
^1
1 m  Below is a list of four statements.  Which state-
     ments is an assertion?
     a.  Sam has long arms.
     b.  Mammal mothers produce milk.
     c.  Birds lay eggs.
     d.  A cactus is not a mammal.
%1
2 m  What characteristic distinguishes a fact?
     a.  It is a statement of truth but may be false.
     b.  It is a truth which can not be objectively
           proven.
     c.  It is something that is a known truth and
           can be proven as true.
     d.  None of the above.
%1
^2
3 t  The antecedent is the "then" potion of an if-then
     rule when a negative action is involved.
%2
4 m  The consequent is
     a.  The "then" portion of an if-then rule.
     b.  Executed when the antecedent portion of a
           deduction system is satisfied.
     c.  Both a and c.
     d.  Neither a or c.
%3
5 t  A reaction system has actions contained in the
     antecedent portions of its rules.
%4
^3
6 m  A rule is triggered when
     a.  All of the if patterns are satisfied.
     b.  All of the consequent patterns are satisfied.
     c.  The rule is fired.
     d.  None of the above.
%5
7 y  When more than one rule is triggered in a reaction
     system a conflict-resolution procedure is needed
```

```
     to decide which rule is to be used.
%6
^4
8 t  When backward chaining, the process moves from a
     rule's consequent to the rule's antecedent.
%7
****
```

File p1c7.a

```
1 a
2 c
3 false
4 c
5 false
6 a
7 yes
8 true
****
```

File p1c7.m

```
1     An ASSERTION is a statement that something is a
   fact.  An assertion may be false.  Assertions and
   facts are different concepts.  A FACT is something
   that is known to be true.  A fact can not be false.
   A set of assertions is called a WORKING MEMORY.
++++
2     The rules of a rule-based system are represented
   in IF-THEN statements.  When the THEN portion of the
   rules specify new assertions to be added to the
   working memory the system is a DEDUCTION SYSTEM.  If
   the THEN portions contain actions the system is a
   REACTION SYSTEM.
++++
3     FORWARD CHAINING is the problem-solving process
   of working from the IF patterns (the ANTECEDENTS)
   TO the THEN patterns (the CONSEQUENTS).  The if
   patterns identify the situations to deduce a new
   assertion or to perform an action.
++++
4     BACKWARD CHAINING begins with a set of HYPOTHESES
   and the antecedent-consequent rules.  The process
   tries to match each hypothesis with the CONSEQUENT of
   a rule.  If there is a match, the process tries to
   support the match by matching each of the rule's
   antecedents by matching to the assertions stored in
   working memory.  Or it may need to create a new
   hypothesis then backward chain through to the next
   rule.
****
```

File p1c7.exm

```
1     A FACT is something which is true or has actually
   happened.  An ASSERTION is something that is stated
   positively but has no objective proof that it is true.
   In the previous question, it is undeniably known
   that mammals give milk, birds lay eggs and cacti are
   plants not animals.  These are facts.  "Sam has long
   arms." is stated positively but must be taken on faith
   that it is true unless you known Sam personally.  This
   is an assertion.
++++
2     An ANTECEDENT is the IF pattern portion of a
   DEDUCTION SYSTEM rule.  Antecedent means comes before
   or precedes something.  The if portion of a rule
   (the antecedent) must be satisfied before the then
   portion can be executed.
++++
3     The THEN pattern portion of a DEDUCTION SYSTEM
   rule is called the CONSEQUENT.  This is the part that
   happens as a result when the if antecedents are
   satisfied.  The consequent contains new assertions
   to be inserted into the WORKING MEMORY.
++++
4     The THEN portion, the CONSEQUENT, of a REACTION
   SYSTEM contains actions to be carried out when the
   antecedent is satisfied.
++++
```

```
5     In FORWARD CHAINING  a rule is TRIGGERED (or
   passes to the then patterns) when all of the patterns
   in the if portion are satisfied (deemed true).  The
   rule is FIRED when a action is performed or a new
   assertion is established.
++++
6     A DEDUCTION SYSTEM usually fires all triggered
   rules.  A problem occurs when more than one rule
   is triggered at one time.  A CONFLICT-RESOLUTION
   PROCEDURE must be used to decide which rule should
   fire.
++++
7  An example of BACK CHAINING:
     Hypothesis:  Sam is a cat.
     Antecedent / Consequent      |   Working Memory:
     Rule:
     IF Sam has a long tail.      |   Has a long tail.
     and
     Sam has a sleek body.        |   Has a sleek body.
     THEN
     Sam is a cat.                |

      The hypothesis matches the consequent of the
   rule.  Each of the if parts are compared with the
   assertions in the working memory.  Since each
   matches the hypothesis is true.
****
```

File p1c7.exq

```
^1
1 y  An assertion is something that is stated as true
     but may be a falsehood.
^2
2 t  The important thing to remember about facts is
     that facts are things that are known to be true.
^3
3 m  The "if" portion of an "if-then" rule is called
     the
     a.  Consequent
     b.  Deduction portion
     c.  Antecedent
     d.  None of the above.
^4
4 y  When the antecedent of a deduction system is sat-
     isfied, the assertions contained in the consequent
     are inserted into the working memory.
^5
5 m  The consequent of the rule:

       IF the object is red           AND
       the object grows on a tree     AND
       the object is an apple
       THEN  pick the object.

     a.  Is not present here.
     b.  Is the "then" part.
     c.  Is part of the rule in a reaction system.
     d.  Answers b and c.
^6
6 t  A rule is fired when all of the "if" patterns
     are satisfied.
^7
7 m  A conflict-resolution procedure
     a.  Decides which triggered rule to fire when
           more than one is triggered at one time.
     b.  Could use a weighted rule scheme to decide
           which simultaneously triggered rules to fire.
     c.  Decides which rule to trigger in a deduction
           system.
     d.  Answers a and b.
^8
8 m  For a hypothesis to be true
     a.  It must match the consequent of a rule.
     b.  Have all of the rule's antecedents match the
           assertions in working memory.
     c.  May need to backward chain through more than
           one rule.
     d.  All of the above.
****
```

File p1c7.exa

```
1 yes
2 true
3 c
4 yes
5 d
6 false
7 d
8 d
****
```

Chapter 9

File p1c9.int

```
-CHAPTER 9
-Frames and Inheritance

     In Chapter 9 you will learn what a FRAME is and
what its SLOTS and SLOT VALUES are.  You will learn the
basic ideas and the procedures associated with FRAME-
REPRESENTATION.  Chapter 9 also teaches you the impor-
tant part played by INHERITANCE in problem-solving.
****
```

File p1c9.q

```
^1
1 t  An instance frame describes a particular thing.
%1
^2
2 m  If a frame has a AKO slot it
     a.  Is a member of a class.
     b.  Is a direct subclass.
     c.  Can not be a member of a class.
     d.  None of the above.
%2
^3
3 y  An instance constructor takes as input the name
     of the class it belongs to and outputs an instance
     automatically connected by an Is-a slot to that
     class.
%3
^4
4 y  The instance My_Car is connected to the class
     Escort by an Is-a slot.  The class Escort is con-
     nected to the class Ford by a AKO slot.  The class
     Ford has a slot with a value of "has ford decal".
     Does the instance "My_Car" have this characteristic?
%4
^5
5 m  A class-precedence list:
     a.  Determines which when-constructed procedures
           to use.
     b.  Is constructed such that each class appears
           before any of its superclasses.
     c.  Is constructed so that a class' direct super-
           class appears before any other direct super-
           class to its right.
     d.  Answers a, b and c.
%5
^6
6 t  When-read procedures are seen only when instances
     are created and the newly created slots are filled.
%6
7 m  A frame system includes:
     a.  Demon Procedures
     b.  Constructors, writers, and readers
     c.  Frames and slots
     d.  All of the above.
%7
****
```

File p1c9.a

```
1 true
2 b
3 yes
```

```
4 yes
5 d
6 false
7 d
****
```

File p1c9.m

```
1     A FRAME is similar to the node of a semantic net.
   A frame's SLOTS corresponds to the links coming out
   from a node in a net.  SLOT VALUES are similar to
   the destination of a link emanating from a node.  A
   FRAME then is a node and all of the links coming out
   from or emanating from the node.
++++
2     INSTANCES belong to CLASSES unless they are
   exceptions.  An Is-a (is-a-member-of) slot identifies
   an instance frame as being a member of a particular
   class.  A CLASS is tied to another class through a
   AKO (a-kind-of) slot.
++++
3     ACCESS PROCEDURES are used to create and manipu-
   late instances and classes.  CONSTRUCTORS create
   instances or classes.  A SLOT-WRITER gives the slots
   their values.  A SLOT-READER outputs a slot value
   when the frame and the name of a slot is input.
++++
4     INHERITANCE, when applied to frames, means that a
   frame acquires the slot values of the frames it is
   connected to by Is-a slots and AKO slots.  It
   inherits the slot values from its direct class(es)
   and superclass(es).
++++
5     A CLASS-PRECEDENCE LIST is an ordered list of
   classes.  It is formed by following the AKO links of
   the classes which an instance belongs.  In a
   branching hierarchy, a depth-first search pattern is
   used to order the classes.
++++
6     DEMON PROCEDURES are procedures which are
   undetected until they are requested.  Such procedures
   include WHEN-REQUESTED, WHEN-READ and WHEN-WRITTEN
   procedures.  These procedures wait around until they
   detect a request, a read, or a write operation
   respectively.
****
```

File p1c9.exm

```
1     A FRAME may describe an individual thing such as
   a particular cat named Sam.  This is an INSTANCE
   FRAME or INSTANCE.  A frame may describe a group of
   things, a class.  This type of frame is a CLASS FRAME
   or a CLASS.
++++
2     AKO stands for "a-kind-of".  This slot signifies
   that this CLASS frame is a member of the class named
   in the slot value.  Similarly, IS-A means "is-a-
   member-of-the-class".  This slot signifies the frame
   is a member of the class in the slot value.
++++
3     An INSTANCE CONSTRUCTOR creates an instance frame.
   It's input is the name of the class it is a member
   of.  The result is an instance which is automatically
   connected to the designated class by an Is-a slot
   in the new instance.
++++
4     An instance INHERITS all of the properties and
   characteristics of all of its superclasses.  That
   means all classes it is connected to directly and
   indirectly through Is-a slots and AKO slots.  For
   instance, the instance Elm inherits all of the
   characteristics of the class Deciduous (connected
   by an Is-a slot) and the class Trees (connected
   to Deciduous by an AKO slot).
++++
5     A CLASS-PRECEDENCE LIST illustrates the class
   hierarchy.  A class or instance is followed by each
   of the class' or instance's superclasses.  A
   BRANCHING class hierarchy contains more than one
   Is-a or AKO links above a class.  One way to order
```

```
   the class hierarchy into a precedence list is to
   use depth-first search.
++++
6     The WHEN-READ PROCEDURES are apparent only when
   the associated instance or class slot is read.
   Otherwise it lirks around until the action occurs.
   It is a DEMON PROCEDURE.
++++
7     A FRAME SYSTEM is a semantic net representation
   which uses frames and slots instead of nodes and
   links.  It uses AKO links to build class hierarchies
   and Is-a links to establish instance class member-
   ship.  DEMON PROCEDURES supply default values, over-
   ride slot values, and maintain constraints.  CON-
   STRUCTORS, WRITERS and READER procedures are also
   included.
****
```

File p1c9.exq

```
^1
1 m  A frame which describes an individual thing is
     called a
     a.  Individual frame
     b.  Instance frame
     c.  Class frame
     d.  Class individual frame.
^2
2 m  The frame Jack has an Is-a link.  Which value
     would best fit in this slot.
     a.  Boy Scout
     b.  Human Being
     c.  Lives in a house
     d.  Can't really tell.
^3
3 t  An Is-a slot is used by an instance constructor
     to connect a newly created instance to another
     instance.
^4
4 t  An instance frame acquires all of the charac-
     teristics from all of the superclasses above it
     through inheritance.
^5
5 t  A class-precedence list orders a class' super-
     classes from left to right in a branching hier-
     archy.
^6
6 m  A demon procedure
     a.  Stays unseen until a request, a read, or a
           write operation occurs.
     b.  Causes much mischief and grief when it is
           activated.
     c.  Neither of the above.
     d.  Both a and b.
^7
7 y  A frame system includes all of the components
     discussed in Chapter 9 including frames and their
     slots, demon procedures and constructors, readers,
     and writers.
****
```

File p1c9.exa

```
1 b
2 a
3 false
4 true
5 true
6 a
7 yes
****
```

File p1c13.int

```
-CHAPTER 13
-Logic and Resolution Proof

     The concise and universally understood ideas of
logic are important tools in problem-solving paradigms.
Chapter 13 will present the notations used in logic and
its inference rules such as MODUS PONENS, MODUS TOLENS
and RESOLUTION.  You will also learn about proofs and
using PROOF BY REFUTATION and the RESOLUTION THEOREM
PROVING.
****
```

File p1c13.q

```
^1
1 m  In logic, antecedent-consequent rules are written
     as _________.
     a.  If-then statements
     b.  Predicates
     c.  Truth tables
     d.  Arguments
%1
2 m  A logical connective
     a.  Joins two predicates.
     b.  Is either an 'OR', a 'NOT', and 'AND' or an
           '=>'.
     c.  Neither a nor b.
     d.  Both a and b.
%2
3 y  A conjunction joins two predicates using "&" (AND).
%3
4 m  A disjunction is
     a.  Two predicates joined by "V" (OR).
     b.  A set of unjoined predicates.
     c.  Two predicates joined by "&" (AND).
     d.  Two predicates that must both be true for the
           the disjunction can be true.
%4
^2
5 t  A truth table can be used to determine the true or
     false value of a complex expression.
%5
^3
6 t  De Morgan's laws are illustrated by the
     expressions:

     E1 & (E2 V E3) <=> (E1 & E2) V (E1 & E3)
     E1 V (E2 & E3) <=> (E1 V E2) & (E1 V E3)
%6
^4
7 t  An expression including a ForAllx can be proved
     false if even one x is found that makes the
     expression false.

     Is the expression,
       ForAllx[Wings(x) => Birds(x)]
     true or false?
%7
8 y  If a single object (x) can be found to make a
     particular ThereExists expression true the
     entire expression is true.

     Is the expression,
       ThereExistsx[Wings(x) => Bird(x)]
     true?
%8
^5
9 t  Only a term can be an argument to a predicate.
%9
10 m Which of these are well-formed formulas?
     a.  Feathers(Sams)
     b.  Feathers(Sam) & Bird(Sam)
     c.  ForAllx[Feathers(x) => Bird(x)]
     d.  All of the above.
%10
^6
11 m A clause is
     a.  A wff consisting of a disjunction of literals.
```

```
     b.  A predicate and its arguments.
     c.  An atomic formula.
     d.  A term.
%11
^7
12 m Modus ponens states that
     a.  If S1 => S2 and ~S2 that ~S1 logically follows.
     b.  If S1 => S2 and S1 then S2 logically follows.
     c.  If (S1 => S2) & (S2 => S3) then (S1 => S3).
     d.  None of the above.
%12
13 y If (Greenleaves(Maple) V (Tall(Maple)) is true
     and (~Tall(Maple) V Trunk(Maple)) is true then
     is (Greenleaves(Maple) V ~Tall(Maple)) true?
%13
^8
14 y Theorem A is to be proven.  If theorem A's negation
     (~A) is proven to be true, is theorem A also true?
%14
^9
15 m In step 1 all implications are removed.  Which
     choice illustrates step 1?
     a.  Hard(x) -> Rock(x) becomes ~Hard(x) V Rock(x)
     b.  Hard(x) & Rock(x) becomes ~Hard(x) V ~Rock(x)
     c.  ~(Hard(x) & Rock(x)) becomes ~Hard(x) V ~Rock(x)
     d.  None of the above.
%15
^10
16 y Step 2 moves the negatives of an expression down
     to its atomic formulas.
     For this expression:
       ~ForAllx[(Greenleaves(x) & Tall(x))]
     is this the correct result?
       ThereExistsx[~Greenleaves(x) V ~Tall(x)]
%16
^11
17 t A skolem function removes the existential quant-
     ifiers by renaming the variables of the quantifiers.
%17
^12
18 y Renaming variables eliminates confusion when the
     the variable appears more than once in an
     expression.
%18
^13
19 y In this expression,
       ForAllx[Greenleaves(x) & ForAlly[Tall(y) &
            Cones(y)]]

     moving the universal quantifiers results in:
       ForAllxForAlly[Greenleaves(x) & Tall(y) &
            Cones(y)].
%19
^14
20 t Moving the disjunctions down to the literal means
     using de Morgan's laws to make the change.
%20
^15
21 m Which choice correctly eliminates the conjunctions?
     ForAllx[(E1(x) & (E2(x) V E3(x))) & (E1(z) V E2(x))]

     a.  ForAllx[E1(x) V (E2(x) V E3(x)) V
           (E1(z) V E2(x))]
     b.  ForAllx[E1(x)]
         ForAllx[E2(x) V E3(x)]
         ForAllx[E1(z) V E2(x)]
     c.  E1(x), (E2(x) V E3(x)), (E1(z) V E2(x))
     d.  There are no changes to be made.
%21
^16
22 m When step 7 was applied to an expression its
     result was:
     ForAllx[E1(x) V E2(x)]
     ForAllx[E1(x) V E4(x)].
     What must now be done?
     a.  Nothing else must be done.
     b.  Rename the x variable of E1().
     c.  Rename the x variables of E2() and E4().
     d.  Rename the E1 variable x and remove the
           universal quantifiers.
%22
****
```

File pic13.a

```
1 b
2 d
3 yes
4 a
5 true
6 false
7 false
8 yes
9 true
10 d
11 a
12 b
13 no
14 no
15 a
16 yes
17 false
18 yes
19 yes
20 false
21 b
22 d
****
```

File pic13.m

```
1     In logic, INFERENCES (if-then statements) are
   expressed in PREDICATES.  More than one predicate
   can be combined into CONJUNCTIONS and DISJUNCTIONS
   using LOGICAL CONNECTIVES.
++++
2     TRUTH TABLES are used to show the values possible
   for various argument combinations.  For instance,
   for the conjunction (E1 & E2) this truth table applies

   |E1 |E2 ||(E1 & E2)|        Both E1 and E2 must be
   |---|---||---------|      true for the conjunction
   | T | T ||    T    |      to be true.
   | T | F ||    F    |
   | F | T ||    F    |
   | F | F ||    F    |
++++
3     Logical connectives have several useful properties.
   These are COMMUTATIVE, DISTRIBUTIVE, ASSOCIATIVE,
   DE MORGAN'S LAWS and when 2 negations are present they
   cancel each other out (the value is positive or true).
++++
4     QUANTIFIERS tell you when specific expressions are
   true.  The UNIVERSAL QUANTIFIER (The symbol is an
   inverted A but here we will use the word ForAll.) says
   that if the expression is true then any object sub-
   stitution replacing x makes the expression true.
      The EXISTENTIAL QUANTIFIER (The symbol is a back-
   wards E but here we will use the word ThereExists.)
   says that there is at least one argument x somewhere
   that will make the expression true.
++++
5  Logic's vocabulary:

   TERMS:              objects, variables, functions
   ATOMIC FORMULAS:    individual predicates with their
                       associated arguments
   LITERALS:           atomic formulas and negated atomic
                       formulas
   WELL-FORMED
   FORMULAS (wffs):    1. literals
                       2. wffs connected by ~, &, V or =>.
                       3. wffs surrounded by quantifiers

    NOTE:  Here " ~ " is used to symbolize negation!
++++
6  More vocabulary:

   SENTENCE:          A wff whose variables are inside
                   the corresponding quantifier's
                   scope.
   BOUND VARIABLE:    A variable found within the scope
                      of a quantifier.
   FREE VARIABLE:     A variable that is not bound.
   CLAUSE:            A wff made up of a disjunction of
                      literals.
```

```
++++
7  The RULES OF INFERENCE are used to prove THEOREMS.

   MODUS PONENS:  If there is an AXIOM, (E1 => E2) and
                  another axiom of the form, (E1),
                  the E2 logically follows.
   RESOLUTIONS:   If there is an axiom, (E1 V E2) and
                  another axiom, (~E2 V E3) then
                 (E1 V E3) logically follows.
                 i.e. E2 and ~E2 cancel each other
   MODUS TOLENS:  If there is an axiom, (E1 => E2) and
                  another axiom, (~E2), then ~E1
                  logically follows.
   CHAIN RULE:    If (E1 => E2) & (E2 => E3) => E1 => E3
                  and (~E1 V E2) & (~E2 V E3) => ~E1 V E3
++++
8     PROOF BY REFUTATION occurs when it is shown that a
   theorem's negation cannot be true.  This implies that
   the theorem itself is true.
++++
9     To solve harder (more complicated) proofs by
   resolution the initial arbitrary logical expressions
   must be manipulated into clause form.  Nine steps
   are followed in the transformation.

      Step 1  Eliminate implications.
++++
10 Step 2  Move the negations down to the atomic
           formulas.
++++
11 Step 3  Remove the existential quantifiers.
           Use SKOLEM FUNCTIONS.
++++
12 Step 4  Rename variables.  This eliminates confusion
           by renaming any variables with the same
           names.
   For example: if Brick(x) appears twice in an
           expression, renaming the x variable in the
           second occurrence eliminates the confusion
           of two x variables.
++++
13 Step 5  Move the universal quantifiers to the left
           of the expression.  Since the quantifiers
           have uniquely named variables, there is
           no confusion by doing this.
++++
14 Step 6  Move the disjunctions down to the literals.
           The Vs are moved inside the &s.
++++
15 Step 7  Eliminate the conjunctions.
++++
16 Step 8  Repeat step 4 and rename repeated variables.

   Step 9  Eliminate the universal quantifiers.
****
```

File pic13.exm

```
1     A PREDICATE is the way logicians express
   antecedent-consequent (if-then) rules.  A predicate
   is a FUNCTION whose ARGUMENTS are mapped into true
   or false values.
++++
2     LOGICAL CONNECTIVES are used to combine two
   predicates.  Logical connectives include: "OR" (V),
   "AND" (&), "NOT" (here ~) and "IMPLIES" (=>).
++++
3     A CONJUNCTION joins two expressions using "&"
   (AND).  Each expression is called a CONJUNCT. Both
   conjuncts must be true for the entire conjunction
   to be true.
++++
4     A DISJUNCTION joins two predicates using "V"
   (OR).  Each predicate is a DISJUNCT. The disjunction
   is true if both predicates are true or if either
   predicate is true and the other is false.
++++
5     A TRUTH TABLE is an easy way to sort out the value
   of an expressions containing several parts.
   For example:  ((E1 V E2) & (E1 & E2)) V ~E2
   has this truth table.
```

E1	E2	((E1 V E2)	&	~(E1 & E2))	V*	(~E2)
T	T	T	F	T	F	F
T	F	T	T	F	T	T
F	T	T	T	F	T	F
F	F	F	F	T	T	T

* = Column with final results.
The expression is false only when both E1 and E2 are true.

++++

6 DE MORGAN'S LAWS actual distribute the negative symbol (~) over an expression then reverses the logical connective.

de Morgan's Law: ~(E1 & E2) <=> (~E1) V (~E2)
~(E1 V E2) <=> (~E1) & (~E2)

++++

7 The expression: ForAllx[Wings(x) => Birds(x)] is false. To prove that the entire expression is false, a SINGLE x must be found that makes the expression enclosed in brackets false. In this case, if x were a bat the expression is false. A bat has wings but is not a bird. This is all the proof that is needed.

++++

8 ThereExists(x)[Wings(x) => Bird(x)] is, of course, very easy to prove true. Only one object, say x is a robin, is needed to prove that the expression is true. A robin has wings. A robin is a bird. The expression in brackets is true, hence, the entire expression is true.

++++

9 A TERM can be an OBJECT, a VARIABLE ranging over an object, or a FUNCTION. The arguments to predicates can only be terms.

++++

10 A WELL-FORMED FORMULA (wffs) are literals, wffs connected by &, V, or => or ~ (negation) and wffs surrounded by quantifiers.

++++

11 A CLAUSE is a well-formed formula (wff) consisting of literals (atomic formulas and negated atomic formulas) joined in a disjunction (joined with V). An example: Greenleaves(x) V Tall(x).

++++

12 MODUS PONENS states that: If there are two true axioms E1 => E2 and E1 (E2 is to be proven) the E2 is also true. E2 is proven.

++++

13 PROOF BY RESOLUTIONS says that if the two axioms (E1 V E2) and (~E2 V E3) are true, it logically follows that (E1 V E3) is true. The last question,
(Greenleaves(Maple) V Tall(Maple)) is true
(~Tall(Maple) V Trunk(Maple)) is true
so logically
(Greenleaves(Maple) V Trunk(Maple)) is also true.

++++

14 PROOF BY REFUTATION states that if it is proven that a theorem's negation cannot be true, then the theorem is proven true. In the last question, since the negation of A is proven true the theorem cannot be true at the same time.

++++

15 To do step 1, eliminate all implications, you substitute (~E1 V E2) for (E1 => E2).

++++

16 The negations are moved to the atomic formulas by using the identities:
~(E1 & E2) -> (~E1) V (~E2)
~(E1 V E2) -> (~E1) & (~E2)
~(~E1) -> E1
~ForAllx[E1(x)] -> ThereExistsx[~E1(x)]
~ThereExistsx[E1(x)] -> ForAllx[~E1(x)]
Writing these identities out on paper will help you remember the identities and answer the remaining questions.

####

The last question required two identities to get the result. The first moves the negation into the brackets. The second moves the negation to the atomic functions.

```
      ~ForAllx[(Greenleaves(x) & Tall(x))]
    1.  ThereExistsx[~(Greenleaves(x) & Tall(x))]
    2.  ThereExistsx[~Greenleaves(x) V ~Tall(x)]
++++
17    SKOLEM FUNCTIONS eliminate the existential quant-
   ifiers by producing the variable associated with
   the quantifier.

      Example:    ThereExistsx[Hard(x) & Rock(x)]
                  Substitute: skolem function Find(x)
                              for x, then remove the
                              quantifier
                  Result:  Hard(Find(x) & Rock(Find(x)).
++++
18 Here is an example of variable renaming.

      ForAllx[(Tall(x) & Cones(x)) V
              (Tall(x) & Needles(x)]
   Becomes:
      ForAllx[(Tall(x) & Cones(x)) V
              (Tall(z) & Needles(x)]

   By replacing x in the second Tall() with z.
++++
19    Step 5 simplifies the expression by simply moving
   all of the universal quantifiers to the left or
   beginning of the expression.  There is no variable
   confusion because after the renaming step the
   quantifiers have unique variables.
++++
20    To move the disjunctions to the literals, use
   the appropriate distributive law:

      E1 V (E2 & E3) <=> (E1 V E2) & (E1 V E3).
++++
21    Eliminating the conjunctions means rewriting the
   parts of a conjunctions as separate axioms.

            Tall(x) & Cones(x)

   Becomes:  Tall(x)
             Cones(x).
++++
22    When step 7 eliminates the conjunctions and
   rewrites the expression into separate axioms,
   identical variables may become apparent.  This is
   easily handled by renaming the duplicate variables
   as we did in step 4.
      Eliminating the universal quantifiers actually
   means that you realize they are there but just don't
   write them.  All variables are assumed to be
   universally quantified.
****
```

File p1c13.exq

```
^1
1 m  Which of these choices is a predicate?
     a.  If cows have wings then cows can fly.
     b.  Cows can fly!
     c.  Wings(cows)
     d.  Choices a and c.
^2
2 t  The expression:  Hasfur(Cat) & Longtail(Cat)
     has two predicates joined with the logical
     connective "OR".
^3
3 m  Which of these are conjunctions?
     a.  Girl(Roger) & Biologist(Roger)
     b.  Girl(Judy) V Biologist(Judy)
     c.  Girl(Sheri) & ~Biologist(Sheri)
     d.  Answers a and c.
^4
4 t  An "OR" or the symbol "&" join two expressions
     to form a disjunction.
^5
```

```
5 m  Which expression matches this truth table?
     | E1 | E2 || ? | ? | ? |     a.  (~E1 & ~E2) V E2
     -----------------------     b.  (E1 V ~E2) V ~E1
     | T  | T  || T | F | F |     c.  E1 & E2
     | T  | F  || T | F | F |     d.  (E1 V E2) & ~E1
     | F  | T  || T | T | T |
     | F  | F  || F | F | T |
     -----------------------
^6
6 y  Does this truth table illustrate de Morgan's law
     for the expression: ~(B & C) <=> (~B) V (~C) ?

     | B | C || ~ |(B & C)|<=>|(~B)| V |(~C)|
     ---------------------------------------
     | T | T || F |    T    |   | F | F | F |
     | T | F || T |    F    |   | F | T | T |
     | F | T || T |    F    |   | T | T | F |
     | F | F || T |    F    |   | T | T | T |
     ---------------------------------------
^7
7 y  Is the expression:  ForAllx[Tall(x) => Tree(x)]
     true?
^8
8 t  To prove that an existentially quantified express-
     ion is true, all possible values for the variable
     associated with the quantifier must be proven true.
^9
9 m  The argument to this predicate:  Tree(Greenleaves(A))
     a.  Is incorrect.
     b.  Is a predicate
     c.  Is a term
     d.  Answers b and c.
^10
10 m How many wffs are included within this expression?
     ForAllx[(Greenleaves(x) V Tall(x)) & Trunk(x)]
     a.  1              c.  5
     b.  3              d.  2
^11
11 y  Two wffs joined by a "V" is called a clause.
^12
12 y  Given that the two axioms, Feathers(Tweety) and
      ForAllx[Feathers(x) => Bird(x], are true, then is
      Bird(x) also true by modus ponens?
^13
13 t  Proof by resolution generally states that when an
      atomic formula (E2) and its negated form appear in
      the axioms (E1 V E2) and (~E2 V E3) they cancel
      each other  and the remaining expression (E1 V E3)
      is true.
^14
14 y  It has been proven that (~Greenleaves(Maple))
      cannot be true.  Does this mean that
      Greenleaves(Maple) is true?
^15
15 t  The expression:
         Brick(x) => (ThereExistsy[On(x,y) &
                        ~ Pyramid(y)])
      becomes:
         ~Brick(x) V (ThereExistsy[On(x,y) &
                     ~Pyramid(y)])
      in step 1.
^16
16 m  Which choice correctly moves the negations in
      this expression?  ~(Hard(x) & Rock(x))
      a.  (Hard(x) & ~Rock(x))
      b.  (~Hard(x) & ~Rock(x))
      c.  (~Hard(x) V ~Rock(x))
      d.  (~Hard(x) V Rock(x))
^17
17 m  Which wff is the skolem function is this
      expression?
         Greenleaves(Which(x)) V Tall(Which(x))
      a.  Which(x)
      b.  Greenleaves(Which(x))
      c.  Tall(Which(x))
      d.  This expression does not have a skolem
            function.
^18
18 m  Which variable should be renamed?
      (T1(x) & T2(y)) V (S1(y) & T1(x)) V
                 (A1(w) & T1(x))
      a.  y
      b.  x
```

```
      c.  All need to be renamed.
      d.  None need to be renamed.
^19
19 y  Have the universal quantifiers been properly
      moved in this expression?
      ForAllx ForAlly[(E1(x) V E2(y)) &
              (E1(y) V ForAllz[E3(z)]]
^20
20 m  Which choice properly moves the disjunctions for
      this expression?
          ForAllx[S1(x) V (S2(x) & S3(x))]
      a.  ForAllx[(S1(x) V S2(x)) & (S1(x) & S3(x))]
      b.  ForAllx[(S1(x) V S2(x)) & S3(x)]
      c.  ForAllx[S2(x) & (S1(x) V S3(x))]
      d.  There's no need to change the expression.
^21
21 y  The correct result of ;moving the conjunctions
      in this expression:
          ForAllx[(Tall(x) V Cones(x)) & Tall(z) V
                  Needles(x))]
      is:  ForAllx[Tall(x) V Cones(x)]
           ForAllx[Tall(z) V Needles(x)]
^22
22 y  A student finished processing an expression with
      this result.
          (Tall(x) V Cones(x))
          (Tall(z) V Needles(x))
      Is there anything else which must be done?
****
```

File p1c13.exa

```
1 c
2 false
3 d
4 false
5 d
6 yes
7 no
    To prove the expression false, x could be a large
  Saguaro cactus which is tall but not a tree.
8 false
    Only one value for x must be proven true to prove
  that an existentially quantified expression is true.
9 d
10 c
11 yes
12 yes
13 true
14 yes
15 true
16 c
17 a
18 b
19 no
20 a
21 yes
22 no
****
```

PART II

File part2.int

```
-PART II
-Learning and
-Regularity Recognition

     In Part II, you will learn about two types of
learning.  The first is learning by integrating newly
learned information with previously acquired knowledge
Chapter 16 teaches you to analyze the differences in
a series of observations. Chapter 16 also shows you
how computers can use examples to learn.
     The second type of learning is called "database
mining".  In Chapter 22 you will learn about Neural
Nets and how to train them using backpropagation.
****
```

Chapter 16

File p2c16.int

```
-Chapter 16
-Learning by Analyzing Differences

     Chapter 16 teaches you how to learn by analyzing the
differences in a series of observations.  You will learn
about INDUCTION HEURISTICS, including REQUIRE-LINK and
FORBID-LINK HEURISTICS.  Examples and the heuristics
associated with them are presented as methods of learning.
You'll also learn about FELICITY CONDITIONS between
teachers and students.
****
```

File p2c16.q

```
^1
1 m  A computer can "learn" by placing facts into its memory,
     as occurs in a database program, to be accessed later.
     This type of learning is:
     a.  Induction
     b.  Direct Instruction
     c.  Memorization
     d.  Analogy
%1
2 m  Sam has written a program telling his computer how to
     search a file for a given phrase.   What type of
     learning is this?
     a.  Direct Instruction
     b.  Analogy
     c.  Deduction
     d.  Answers a and c
%2
^2
3 t  Artificial Intelligence uses induction and examples to
     teach a computer about something.
%3
4 m  Jenny wanted to teach her computer about cars.  She "told"
     it that cars have 4 wheels, carried passengers and run
     on gas.  Which following example is a good NEGATIVE example
     to use next?
     a.  Has 4 wheels, is red, carries passengers, runs on gas.
     b.  Has 4 wheels, carries large cargoes, runs on gas.
     c.  Has 4 wheels, carries passengers, runs on gas, has a
         trunk.
%3
^3
5 m  A description is changed and enhanced by a series of
     examples.  This description is called a:
     a.  Changing Model
     b.  Near Miss Model
     c.  Evolving Model
     d.  None of the above
%4
^4
6 m  A near miss example is presented and it is missing a link
     found in the evolving model.  What should be done?
     a.  Use the forbid-link heuristic and throw out the example.
     b.  Use the require-link heuristic and change the model's
         link to a Must-be link.
     c.  Try the next example.
     d.  Use the climb-tree heuristic to see if the next higher
         class matches.
%5
7 t  A Must-Not link is placed in a model by the forbid-link
     heuristic when a near miss example contains a link the
     evolving model does not have.
%6
^5
8 t  The climb-tree heuristic generalizes the model when an
     object matches an individual element of a class.
%7
9 y  The model of plants has an Is-a link to "tree".  The
     presented near miss example also has a corresponding Is-a
     link to "bush".  Is it correct to use the enlarge-set
     heuristic to combine these two objects into a new class
     called "woody stemmed plants"?  Hint: bushes do have woody
     stems.
%8
```

```
10 m For the model of "apples", the only colors they may be
     are red, green, and yellow.  Which heuristic should be used
     to evolve the model?
     a.  Require-link
     b.  Climb-tree
     c.  Enlarge-set
     d.  Drop-link
%9
11 m The evolving model of what defines a flower has petal
     number equal to 3.  A near miss example has petal number
     equal to 8.  Which heuristic will be used to change the
     model?
     a.  Close-interval
     b.  Enlarge-set
     c.  Require-link
     d.  Climb-tree
%10
****
```

File p2c16.a

```
1 c
2 a
3 True
4 b
5 c
6 b
7 true
8 false
9 yes
10 d
11 a
****
```

File p2c16.m

```
1     LEARNING is an important property of intelligence.
   It occurs at different levels and in different forms.
   The simplest computer learning is MEMORIZATION: adding
   facts to storage or memory and DIRECT INSTRUCTION:
   telling the computer what to do.
++++
2     INDUCTION is a form of learning which uses EXAMPLES,
   both positive and negative, to teach about a particular
   object or subject.  Slightly varying examples are pre-
   sented.  Each variation teaches a small new fact about
   the object or subject to be learned.
++++
3     An EVOLVING MODEL begins with an initial description.
   The model is gradually changed and augmented by pre-
   senting POSITIVE and NEGATIVE EXAMPLES.
++++
4     A HEURISTIC is a method of finding solutions using
   "rules of thumb", observations and experimentation.
   Six heuristics are used to evolve a model.  Two heur-
   istics SPECIALIZE or restrict the model's description.
   These are the REQUIRE-LINK and the FORBID-LINK heuristics.
++++
5     The last four heuristics GENERALIZE the evolving
   model making it more relaxed, it accepts more, is less
   restrictive.  These four heuristics are the CLIMB-TREE
   and ENLARGE-SET and the DROP-LINK and CLOSE-INTERVAL
   heuristics.
****
```

File p2c16.exm

```
1       To MEMORIZE something a person usually needs
   the information repeated several times unless the
   person has a photographic memory.  A computer has
   a photographic memory and can "memorize" information
   by receiving it then storing it in its memory banks.
   It may come directly from the outside world (through
   scanners, the keyboard, etc.) or by reading files.
++++
2       DIRECT INSTRUCTION requires very little know-
   ledge.  A computer or person simply and blindly
   follows, step-by-step, any instructions presented
   to it.
++++
```

```
3       To illustrate INDUCTION HEURISTICS, let's say
   we want our computer to learn about apples.  First
   we tell it that apples grow on trees, are spherical
   and can be eaten.  Our first example might be a pos-
   itive example of:  grows on trees, can be eaten and
   is red.  The next example might be a negative example
   such as:  grows on trees, is spherical and is orange.
   By this time the computer knows that an apple:  grows
   on trees, is spherical, and is red and NOT orange.
++++
4       An EVOLVING MODEL changes a little at a time
   with each example presentation.  For instance, the
   model of a red apple which grows on a tree and is
   edible, may be slightly changed by including: tastes
   sweet.
++++
5       The REQUIRE-LINK HEURISTICS adds to the model
   characteristics which must be present for an object
   to match the model.  The link appears in the model
   but not in the near miss.  The link is converted to
   a MUST link.
++++
6       The FORBID-LINK HEURISTIC uses near miss examples
   with a link in a place the evolving model does not.
   Here a MUST-NOT link is added.
++++
7       The CLIMB-TREE HEURISTIC climbs a classification
   tree to the next higher common class.  The evolving
   model is relaxed to accept a more general description.
   For example, to build a wall, bricks or stones or
   concrete blocks could be used.  The model can be
   generalized to accept "Blocks" as the object instead
   of the specific elements.
++++
8      The ENLARGE-SET HEURISTICS is used when a class-
  ification tree is not involved.  Objects not commonly
  related may be united into a set.  For example, when
  building a fence, stone blocks or wooded boards or
  heavy wire might be united into the set "building
  materials".  This generalizes the model.
++++
9       The DROP-LINK HEURISTIC is used when no other
  elements can be used.  For example, if only bricks
  and wooden boards can be used to build a fence, the
  Is-a link between the model and the objects is
  removed.  Drop-link is also used when the example
  is missing a link the evolving model has.
++++
10      The CLOSE-INTERVAL HEURISTIC involves numbers.
  For example, when building that fence, the model in-
  cludes 12 inch bricks and example A includes 20 inch
  bricks.  Close-interval expands the model to include
  example A's measurements and assumes any brick from
  12 to 20 inches can be used.
****
```

File p2c16.exq

```
^1
1 t  The simplest way a computer is able to "learn"
     is by putting facts directly into its memory.
     This is similar to memorization in humans.
^2
2 y  Does a computer require a great knowledge to
     perform a task using direct instruction?
^3
3 t  Induction means to use examples, each varying
     slightly from the next, to draw a general con-
     clusion about something.
^4
4 t  A negative example tells something that an object
     must be.
^5
5 t  To change an evolving model, a positive example
     with a small difference can be incorporated into
     the model.
^6
6 m  If the model must have a particular link,
     a.  The link appears in the model but not in the
           near miss example.
     b.  The require-link heuristic is applied.
     c.  The link corresponding link is converted to a
           Must-be link.
```

```
     d.  None of the above answers.
     e.  All of the above answers.
^7
7 3  A model of an apple has the links: grows on trees,
     is red and is edible.  A near miss example has
     links: grows on trees, is red, is edible and is
     used to build fences.  Should the forbid-link
     heuristic be used here?
^8
8 m  The climb-tree heuristic is used when:
     a.  A classification tree is used.
     b.  A set of objects is matched.
     c.  A link points to a different class in the model
           than in the presented example.
     d.  Answers a and c.
     e.  Answers b and c.
^9
9 t  The enlarge-set heuristic is applied when a link
     appears in an example but is not present in the
     evolving model.
^10
10 m The drop-link heuristic generalizes a model when:
     a.  The differing objects between an example and
           the model form an exhaustive set.
     b.  A link contained in the model is absent in
           the presented example.
     c.  Both a and b.
     d.  Neither a nor b.
^11
11 m The close-interval heuristic:
     a.  Closes a set so no other elements may be added.
     b.  Extends a modes's numeric value to include the
           exhibited example's numeric value.
     c.  Removes a model's link to a number value.
     d.  None of the above.
****
```

File p2c16.exa

```
1 true
2 no
3 true
4 false
5 true
6 e
7 yes
8 d
9 false
10 c
11 b
****
```

Chapter 22

File p2c22.int

```
-Chapter 22
-Learning by Training Neural Nets

     In Chapter 22, you will learn about NEURON-LIKE
COMPONENTS which can learn to recognize patterns when
connected in nets.  Included is a review of the prom-
inent properties of real neurons.  You will learn how
to model these properties in neural nets.  Chapter 22
also teaches you how neural nets learn using the
BACK-PROPAGATON PROCEDURE.
****
```

File p2c22.q

```
^1
1 m  The connection of one simulated neuron to another
     corresponds to the __________ connections of a
     biological neuron.
     a.  Neuron-link-neuron
     b.  Synapse-neuron-synapse
     c.  Axon-synapse-dendrite
     d.  None of the above.
%1
```

```
^2
2 m  A neuron is an "all-or-none" device because:
     a.  All of its components must be present for it
           to operate correctly.
     b.  All of the neurons in a net must be influenced
           to respond correctly.
     c.  When the "collective influences" of its inputs
           reach a threshold, it either fires or it stays
           inactive.
     d.  None of the above.
%2
3 t  The weights associated with a simulated neuron,
     simulate the degree of transmission of pulses
     between biological neurons as adjusted by the
     synapses.
%2
4 m  Demon procedures:
     a.  Write the product of the input and weight into
           a synapse's output slot.
     b.  Sum the input / weight products of all the
           synapses.
     c.  Apply the threshold to the product sum to
           obtain the neuron's output.
     d.  All of the above.
%4
5 t  To propagate the output of one neuron to the next,
     a demon procedure multiplies the output of the
     previous node (now the input to this node) and the
     link's weight then sends the product into the next
     neuron.
%5
6 y  The hidden nodes are called hidden because their
     inputs produce no outputs.
%6
7 t  The amount of change in the weights is proportional
     to the amount of good done by individual changes.
%7
8 y  Gradient ascent must use the stair-step threshold
     to be most effective.
%8
9 t  The main idea behind back-propagation is to make a
     large change to a weight only if that change greatly
     reduces the errors in the output layer.
****
```

File p2c22.a

```
1 c
2 c
3 true
4 d
5 true
6 no
7 true
8 no
9 true
****
```

File p2c22.m

```
1    The components of a simulated neuron have many
  similarities to biological neurons.  The axon of a
  biological neuron and links of a simulated neuron
  deliver neural output to the connections of another
  neuron.
++++
2    It is believed that biological learning may occur
  in the narrow gaps between the axons and the dendrites.
  These gaps, called SYNAPSES, influence the firing or
  non-firing activity of the neuron.
++++
3    FEED-FORWARD NETS enable neural nets to recognize
  signal regularity.  DEMON PROCEDURES propagate infor-
  mation from one node layer to the next.
++++
4    The power of a network of simulated neurons, called
  a NEURAL NET, can often be increased by increasing the
  number of layer of neurons.  The layers between the
  input layer and the output layer are the HIDDEN LAYERS
  and contain the HIDDEN NODES.
++++
```

5 The BACK-PROPAGATION PROCEDURE uses hill climbing by making small weight adjustments in the direction of the most improvement. This is GRADIENT ASCENT.
++++
6 The THRESHOLD FUNCTION is called a STAIR-STEP FUNCTION because of the abrupt change between the input and output values. A SQUASHED S THRESHOLD FUNCTION has a gradual change. A smooth curve rather than the sharpness of a set of stair steps.
++++
7 BACK-PROPAGATION adjusts the weights beginning in the final layer of node then works back through the layers to the initial node layer. The weight changes should be in proportion to the reduction of the output error. A large change for a large error reduction.
****

File p2c22.exm

1 In a biological neural system, a signal is passed from one neuron to the next. The AXON passes a neuron's output through a small gap called the SYNAPSE to the DENDRITES of the next neuron.
A simulated neuron has weighted links connecting the nodes. The product of the weights and the neuron's output acts like a synapse determining the nature and strength of the signal. An adder simulates the combined stimulations of a neuron's dendrites. The activity of comparing with a threshold simulates the all-or-none biological neuron operations.
++++
2 The SYNAPSE of a biological makes chemical changes to transfer the nature and strength of the signal (influence) to the next neuron. A simulated neuron has weights associated with its links to another neuron. The nature and strength of the influences of a simulated neuron are adjusted by multiplying the inputs by the link weights.
++++
3 The activation function of a simulated neuron combines the neuron's input and any weights to determine the overall influence of a neighboring neuron.
++++
4 The DEMON PROCEDURES move the information or influences from one neuron to another. One demon moves the signal across the neuron. Another moves the information to the next neuron.
++++
5 One DEMON PROCEDURE acts like the SYNAPSE of a biological neuron. It adjusts the incoming signal's influence by multiplying the input value by the weight of the connection.
++++
6 A neural net may have several layers of nodes to improve its efficiency. The INPUT layer receives the initial signal. The OUTPUT layer transmits the final (output) signal. Between these two layers are the HIDDEN LAYERS of HIDDEN NODES where the learning takes place. The inputs and outputs of these hidden layers are not observable.
++++
7 GRADIENT ASCENT is a movement in the direction of the gradient (in the direction of improvement). In BACK-PROPAGATION, all of the weights are adjusted simultaneously according to how much good is accomplished by the individual weights.
++++
8 The STAIR-STEP THRESHOLD FUNCTION products sharp abrupt changes like the abrupt height changes in a staircase. GRADIENT ASCENT produces small step changes which are lost in the stair-step threshold.
The SQUASHED S THRESHOLD FUNCTION with its smooth transitions better suits the small steps of gradient ascent.
++++
9 To determine whether a particular weight "w" should be changed, the actual output value "o" is subtracted from its desired value "d" to get the OUTPUT ERROR VALUE. A small change, reduction, in the output error should be reflected in a small change in the

```
  weight, a large reduction should produce a large
  change.
****
```

File p2c22.exq

```
^1
1 m  In a simulated neuron, what simulates the
     activity of the biological synapse?
     a.  The addition or combination of all of
           the incoming influences.
     b.  The product of the neuron's link weights
           and outputs of the influencing neuron.
     c.  Comparing the summed products of the
           inputs and weights to the threshold
           function value.
     d.  All of the above.
^2
2 m  The _____ of a simulated neuron corre-
     sponds to the synapse of a biological neuron.
     a.  The weights associated with its links.
     b.  The links between the neurons.
     c.  The threshold values.
     d.  None of the above.
^3
3 t  The activation function sends the combined
     weight / input influence through a threshold
     function.
^4
4 t  A demon procedure moves information through a
     neuron by summing the products of all of the
     inputs and weights and applying a threshold
     function.
^5
5 y  There are several demon procedures in a feed-
     forward net.  One procedure simulates a synapse
     by moving the signal between neurons.
^6
6 t  The layers of neural nodes between the input layer
     and the output layer are "hidden" because their
     outputs are not directly observable.
^7
7 t  The back-propagation procedure begins the changes
     to the connection weights by going back to the
     input layer after data is first propagated through
     the neural net.
^8
8 t  The squashed S threshold is a smooth function
     which better detects the small step changes char-
     acteristic of gradient ascent.
^9
9 m  Back-propagation determines when to adjust a
     particular weight by:
     a.  Deciding if the output error would be reduced
           enough.
     b.  By subtracting the actual output from the
           desired output.
     c.  Both a and b.
     d.  None of the above.
****
```

File p2c22.exa

```
1 b
2 a
3 true
4 true
5 true
6 true
7 false
8 true
9 c
****
```

PART III

File part3.int

```
-PART III
-Vision and Language

     Part III introduces you to visual perception and
language understanding.  You will learn about recog-
nizing objects, describing images, language constraints
and how to express them, and about responding to ques-
tions and commands.
****
```

Chapter 26

File p3c26.int

```
-CHAPTER 26
-Recognizing Objects

     Chapter 26 tackles the problem of recognizing
objects in images.  In Chapter 26, you will learn to
construct templates from stored two-dimensional images
by adding together the weighted values of corresponding
points.  You will be taught methods for determining
which points correspond.  By the end of Chapter 26, you
will know what the LINEAR COMBINATION PROCEDURE is,
how it works and when it is appropriate to use.
****
```

File p3c26.q

```
^1
1 m  Why would people want to have a computer that can
     identify what it sees?
     a.  It makes a good science fiction story.
     b.  In robotics, a computer could choose a specific
           part from many types for assembly-line tasks.
     c.  Hazardous exploration could be carried out by
           robots that could make instant decisions about
           what they observe.
     d.  Answers b and c.
%1
^2
2 y  One of the hurtles to overcome in computer vision
     is the enormous amount of knowledge that is needed
     to interpret each image.
%2
^3
3 m  Shimon Ullman, in 1979, showed that _______ with
     _______ each were nearly enough to find the positions
     of the corresponding vertices.
     a.  4 front views, 3 side views
     b.  3 corresponding vertices, 3 side views
     c.  3 images, 4 corresponding vertices
     d.  Answers a and c.
%3
4 y  Ullman's concept needs only to have some infor-
     mation about the polyhedron's size to determine
     the positions of related vertices.
%4
^3
5 y  Can equations give you the coordinates of vertices
     in a new image if you are given three recorded
     images?
%5
6 m  When an image is projected and rotated to a 90
     degree position relative to the original image, it
     is said to be a _____________ projection.
     a.  Right angle
     b.  Right hand
     c.  Orthographic
     d.  90 degree
%6
^4
7 y  Once alpha and beta are calculated, they can be
     used to predict the x and y coordinates of other
     image points.  These predicted points are compared
```

```
     with actual observed points to identify the
     unknown object.
%7
8 m  A special case can be demonstrated using only two
     model images but:
     a.  Rotation is restricted to around one axis only.
     b.  Three models are needed to handle arbitrary
           rotation, translation and scaling.
     c.  Three models with 4 corresponding points
           identifies an unknown better.
     d.  All of the above.
%8

9 m  The template approach to object identification
     can:
     a.  Handle arbitrary rotation, translation and
           scaling.
     b.  Handles only rotation around a single axis
           well.
     c.  Handle objects with parts and objects with
           curves.
     d.  Answers a and c.
%9
****
```

File p3c26.a

```
1 d
2 yes
3 c
4 yes
5 yes
6 c
7 yes
8 d
9 d
****
```

File p3c26.m

```
1    COMPUTER VISION is the programming of a
   computer to "see" as a human does.  Thus enabling the
   computer to recognize and understand the objects it
   observes.
++++
2    COMPUTER VISION is dedicated to understanding
   images for classification, interpretation, descrip-
   tion and recognition purposes.  There are hurtles
   to overcome, however.  Understanding what an unclear
   image presents is more easily inferred by humans than
   computers.  Obtaining a 3-d picture from a 2-d sensor
   image is another obstacle.
++++
3    IMAGE COMBINATION borrows a concept from geometry
   which combines a front view, side view and top view
   of a polyhedron providing a complete knowledge of
   each vertex's position in a 3-dimensional space.
      The general version, used in object identification,
   says several images, each with a few corresponding
   vertices, will "give you an idea of where those corr-
   esponding vertices are relative to one another in
   three dimensions."  Those several images need not be
   front, side and top views.
++++
4    New images can be created by calculating the 3-
   dimensional coordinates of all of the vertices of
   the polyhedron then projecting the vertices through
   a lens onto a sensor array.
++++
5    One method of object identification is to compare
   the points of an unknown object with those of
   templates stored in a library of images.  Point cor-
   respondences are calculated between the model
   and unknown object.  A search of the model library
   must consider image rotation, scaling and translation.
++++
6    The procedure to identify an unknown object has
   three steps to be repeated until a match is found.
     1.  Find 4 corresponding points in the object and
           library images of the model.
     2.  Use the points to calculate alpha and beta and
           predict x, y values for other image points.
```

3. Compare observed and predicted x, y values for matches.

****

File p3c26.exm

1 Teaching a computer to understand what it "sees" is very important for a wide variety of reasons. As the last question implies, robotics heavily depends on identifying objects and understanding what they are. Consider a "seeing-eye" robot of the future which is able to sense, interpret then transmit signals to a blind person's brain about the environment around them.

++++

2 Images are divided into tiny pieces called PIXELS. There are 24 bits associated with each pixel. For a high resolution image, 4096 pixels by 4096 pixels there is 402,653 bits of information. This is certainly an obstacle when many repetitive operations are performed on each pixel and the knowledge needed to interpret the data.

++++

3 For many years experts believed object identification processing had to take place on three separate levels. Later it was found that a few images with corresponding vertices identified could produce an identification. Later still, Shimon Ullman determined the number of images (3) and the number of corresponding vertices (4) needed for object identification.

++++

4 Shimon Ullman found that 3 images with 4 corresponding vertices can be combined to make an object identification. The only information missing is the size of the object (polyhedron) which can be determined from the distance between any two of the object's vertices.

++++

5 Luckily, there are simple equations that can calculate the coordinate values of a projected image using the coordinate values of three stored images. It is then possible to determine all of the new image's parameters.

++++

6 The dictionary meaning of ORTHOGRAPHIC is "of right angles and perpendicular lines". Hence, rotating an image 90 degrees (a right angle = 90 degrees) is the orthographic projection image of the original image.

++++

7 The corresponding points of the three library model images are used to predict the point values of a projected image. The predicted point values are then compared with the values of points in the observed object. If all of the points values match the identification is confirmed.

++++

8 Using the corresponding points of THREE model images allows identification of objects which have been rotated (in any direction), translated (moved to another place), or scaled (changed size). Two images are enough if only rotation around one axis is allowed.

++++

9 When templates are stored in a library model, the possibility of the object being rotated (turned), scaled (size changed), or translated (moved) are considered and templates are stored reflecting these possibilities.

****

File p3c26.exq

^1

1 t Through Artificial Intelligence and Computer Vision robots may be able to make unsupervised trips to the great depths of the oceans to retrieve items from sunken ships.

^2

2 m Each pixel of an image has a value
a. Containing one bit of information.

```
     b.  Containing 24 bits of data of color and
         intensity information.
     c.  Containing 48 bits of information.
     d.  Containing 24 bits of random data.
^3
3 t  Shimon Ullman determined that 2 images with 4
     corresponding vertices is enough to find the
     relative positions of the vertices.
^4
4 t  To obtain all the knowledge about a polyhedron
     you need 3 images with 4 corresponding vertices
     and some information about the size of the
     polyhedron.
^5
5 m  Three recorded images can give you a forth
     image if
     a.  The fourth image is recorded and stored.
     b.  You are given some information of the object's
           size.
     c.  You derive the vertice's coordinates and
           use an equation to derive the fourth image's
           corresponding points.
     d.  Answers b and c.
^6
6 m  An example of an orthographic projection of a
     facial view of Abraham Lincoln's bust is
     a.  An image of the right side of his head and
           the right ear.
     b.  A full facial view.
     c.  A full view of the back of his head.
     d.  A view of his right hand.
^7
7 t  To identify an unknown object, the points of the
     unknown object which may correspond to those of
     the library images, are compared to calculated
     corresponding points of a predicted image.
^8
8 t  Three images are needed to identify an object
     which has been translated.
^9
9 y  An object image which has been rotated 180 degrees
     and reduced to half its original size can be
     identified using templates in a model library.
****
```

File p3c26.exa

```
1 true
2 b
3 false
4 true
5 d
6 a
7 true
8 true
9 yes
****
```

Chapter 27

File p3c27.int

```
-Chapter 27
-Describing Images

     Chapter 27 teaches more about computer vision by
examining two problems.   BINOCULAR STEREO PROBLEM,
the first problem, uses two slightly different view-
points to find the distance from surfaces to the viewer.
The second problem, SHAPE-FROM-SHADING PROBLEM, uses
the way object surfaces are shaded to determine the
surface-direction at each point.
****
```

File p3c27.q

```
^1
1 m Edge isolation, in reality, is not a simple
    straight forward procedure because :
```

```
     a.  The sensor may have sensing errors, elec-
           tronic noise, and/or a brightness detection
           limitation.
     b.  Technology is not yet advanced enough to detect
           edges.
     c.  The image may have rounded edges, dust, finger-
           prints, and/or mutual illumination effects.
     d.  Answers a and c.
%1
^2
2 m The four step technique described in the textbook
    helps detect edges in images by:
    a.  Selecting image points and averaging their
          neighboring brightness points.
    b.  Finding the average of the left-neighbor, right-
          neighbor differences.
    c.  Examining resulting values for edge-signaling
          value combinations.
    d.  All of the above.
%2
3 y The point-spread function shows the influence of
    a single input image point on the output image.
%3
^3
4 m  Given the values:                    |    P     | -----
        focal length = 2                  |   /|\    |   d
        alpha = 2, beta = 3               |  / | \   |
     find the distance from               |L   |   R | -----
     the sensors to the object      /     |    |     |\  f
     being viewed.              ---/---|---|----|---|-\----
                                 /  a   | l |  r |  b  \
     a.  4.2
     b.  6.2
     c.  1.35
     d.  None of the above.
%4
^4
5 m  A Lambertian surface
     a.  Reflects no light so it is black.
     b.  Appears mottled because of an uneven surface.
     c.  Is equally bright from all viewpoints.
     d.  Reflects all light so appears white.
%5
^5
6 t  A surface's albedo is its degree of smoothness.
%6
7 t  A reflectance map has isobrightness lines to
     indicate degrees of surface brightness.
%7
****
```

File p3c27.a

```
1 d
2 d
3 yes
4 a
5 c
6 false
7 true
****
```

File p3c27.m

```
1    Identifying an object often means isolating the
  edge between two flat surfaces.  This seems to be a
  simple task because of the sharp change in brightness
  from one surface to another. Actually, that sharp
  change is usually corrupted by imperfections in the
  object itself and the inconsistencies of the imaging
  equipment.
++++
2    A four step averaging technique can be used to
  smooth noise-corrupted steps produced by the object
  and imaging imperfections.
     The four steps involved can be combined into an
  averaging procedure, a POINT-SPREAD FUNCTION.  It
  shows the spreading influence of a single nonzero
  brightness point in an output image.
++++
3    STEREO VISION uses two "eyes" to determine
   distance using the formula
```

```
      d = fb / alpha + beta.          |    P    | -------
                                      |  /| \   |    d
      d = distance,                   |/  |   \ |
      f = focal length,               L   |     R -------
      bl = baseline = l + r     /     |   |     | \  f
      a = alpha              --/---|---|---|---|---\----
      b = beta               /  a  | l |   r |   b  \
                                   |<- bl ->|
```

++++

4 The surface direction (the direction the surface lies in with regard to the light source) can be determined by knowing the location of the light source and the brightness of the surface. The brightness of a surface depends only on the direction of the light source.

++++

5 To find the observed brightness(E) of the surface, the incident (i) angle's cosign is multiplied by the surface albedo (p). E = pcosi.

```
sun                 |           viewer  i = incident angle
 \                  |<-sn        /      e = emergent angle
  \                 |           /       g = phase angle
   \ <-i->          | <-e->    /        sn = surface normal
    \               |       /
     \ <- g -> /    |      /
      \        /    |     /
       \|/          |
```

****

File p3c27.exm

1 Finding the edge between two flat surfaces of an object is made difficult by brightness sensitivity variations, electronic noise, sensor limitations, and other things produced by the imaging equipment. The object itself can add complications by having curved edges, scratches, shadows, fingerprints, or dust to obscure the edge definition.

++++

2 The image averaging technique is used to accentuate edge differences. Various image points (I[i]) are selected. The average brightness of its neighboring points is calculated and stored in an array. Step two finds the AVERAGE-FIRST-DIFFERENCE of the left-neighbor brightness and the right-neighbor brightness in the first array.

```
    F[i] = A[i+1] - A[i-1] / 2
```

These values are stored in a second array. Step two is repeated using the second array to obtain the AVERAGE-SECOND-DIFFERENCE array.

```
    S[i] = F[i+1] - F[i-1] / 2
```

This last array is examined for edge-signaling combinations of peaks, steep slopes and zero crossings.

++++

3 The POINT-SPREAD FUNCTION combines the four steps of difference and averaging into a single formula,

```
    O[i] = summation P[j-i] X I[j]
```

It shows the influence of a brightness point on an output image.

++++

4 The equation to determine the distance from the sensors to the object is d = (f x b) / (alpha + beta).

d = distance,
f = focal length,
b = baseline = l + r
L = left lens
R = right lens

```
                                |    P    | --------
                                |  /| \   |    d
                                |/  |   \ |
                                L   |     R --------
                              /     |   |     | \  f
                           ---/---|---|---|---|---\-----
                            /  a  | l |   r |   b  \
```

++++

5 LAMBERTIAN SURFACES are also called perfect diffusers, matte or nonspecular. These surfaces reflect light equally all over the surface. The brightness is the same from any point its viewed from.

++++

6 ALBEDO is defined in Webster's Dictionary as the reflecting power of a planet or satellite. The surface albedo then is essentially its brightness.

++++

```
7     A REFLECTANCE MAP is the brightness of a sphere
   projected on to a plane.  ISOBRIGHTNESS LINES are
   lines on a sphere or reflectance map along which
   the surface brightness is a constant value.
****
```

File p3c27.exq

```
^1
1 t Edge isolation is always easy because the edge
    between two surfaces is a distinct well defined
    line.
^2
2 y This method of averaging differences smooths the
    inconsistencies of noise-corrupted brightness
    detection.
^3
3 t The point-spread function shows how a nonzero
    brightness point spreads its influence in an output
    image.
^4
4 y Stereo vision finds the distance from a pair of
    sensors to an object by dividing the focal length
    (f) times the baseline (b) by alpha (a) plus
    beta (b).
^5
5 t A surface that appears to have the same brightness
    from all points of view is a perfect diffuser or
    a Lambertain surface.
^6
6 m The albedo of a surface depends on
    a.  The ;material the surface is made of.
    b.  Is constant for a particular material.
    c.  Both a and b.
    c.  Neither a nor b.
^7
7 m Given this reflectance map: .A(.7)          .F(.4)
    where would the          .B(.5)       .G(.7)
    isobrightness                 .C(.6)          .E(.7)
    line go.                               .D(.65)

    a.  From A to G to E
    b.  From B to C to D
    c.  From C to D to E
    d.  From C to D to G
****
```

File p3c27.exa

```
1 false
2 true
3 true
4 yes
5 true
6 c
7 a
****
```

Chapter 28

File p3c28.int

```
-CHAPTER 28
-Expressing Language Constraints

     Chapter 28 teaches you about language constraints
and how to express them in SENTENCE TREES.  You will
learn about X-BAR SCHEMA and how it is used to build
the sentence trees.  By the end of Chapter 28 you will
understand the difference between LINGUISTIC COMPETENCE
and LINGUISTIC PERFORMANCE.
****
```

File p3c28.q

```
^1
1 m Linguists often use _____ to formulate
    linguistic constraints.
```

```
    a.  Sentence pairs
    b.  Neural Nets
    c.  Antecedents and consequents
    d.  None of the above
%1
^2
2 t Phrases form around specific words which give the
    phrase its name.
%2
^3
3 m Binary trees
    a.  Are seldom used to describe the parts of
          a phrase.
    b.  Best illustrate the properties shared by all
          types of phrases.
    c.  Lack the branching flexibility to illustrate
          the many types of phrases.
    d.  None of the above
%3
^4
4 t A binary tree description of a particular sentence
    shows a branch descending from the NP node and to
    the left of the noun.  This branch of word(s) is
    called the complement of the noun.
%4
^5
5 t An X-bar binary tree can only be used to diagram
    verb phrases.
%5
^6
6 y Does an inflection phrase inflect a verb by adding
    tense to indicate when an action took place?
%6
^7
7 m A complementizer phrase
    a.  Is a phrase that describes a person, place or
          thing in a favorable light.
    b.  Is part of a noun phrase.
    c.  Handles embedded sentence-like groups and can
          describe the relationship between two similar
          sentences.
    d.  None of the above.
%7
****
```

File p3c28.a

```
1 a
2 true
3 b
4 false
5 false
6 yes
7 c
****
```

File p3c28.m

```
1     Linguists look for simple, concise and compre-
   hensive explanations for how language is developed
   and understood.  They use various devices to under-
   stand formulate the constraints of language such
   as SENTENCE PAIRS.
++++
2     Sentences often have groups of words (phrases)
   which naturally seem to go together.  The phrases
   have central words which give the phrase its name
   through the type of the word.  For instance, this
   prepositional phrase forms around the word "pot".

       "in the large round pot."
++++
3     Linguists use sentence trees to show how the parts
   of a phrase fit together.  Many linguists feel that
   binary trees are a simpler but a richer way to des-
   cribe word relations.
++++
4     Phrases of all types, NOUN PHRASES, VERB PHRASES
   and PREPOSITIONAL PHRASES, can have the same
   structure when diagramed in a binary tree.
++++
```

5 A BINARY X-BAR TREE is a generalized binary tree representation. There are three types of nodes. XP nodes correspond to the NP node of a noun phrase. SPECIFIERS enter from one side of an XP node. COMPLEMENTS enter X BAR nodes from the side opposite to the specifier. The last node is an X node, the head node.
++++
6 A INFLECTION PHRASE adds a sense of time to a sentence. TENSE is added, for instance, by ending the verb in '-ed' for past tense or using 'will' for the future tense, the phrase is inflecting the verb.
++++
7 The COMPLEMENTIZER PHRASE shows embedded sentence-like phrases. Linguists often use complementizer phrases to show the relationship between different sentence types.
****

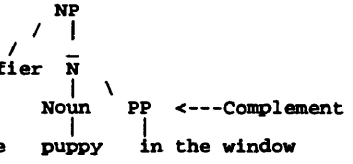
File p3c28.exm

1 To understand linguistic constrains, linguists often use SENTENCE PAIRS. Each sentence pair has one correct or natural sounding sentence. The other sentence is a near miss with something slightly wrong. For instance, one word may have the wrong tense or may be the wrong pronoun.
++++
2 A sentence, when examined, has small groupings of words which seem to fit together. For example, in the sentence,

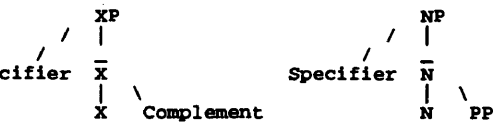
"The red fish swims at the bottom of the fishbowl.", three word groups are formed: - The red fish,
- at the bottom, and
- of the fishbowl.

The phrases form around a word whose type gives the phrase its name. "At" and "of" are prepositions and thus the phrases they are in are called PREPOSITIONAL PHRASES. "The red fish", is a noun phrase.
++++
3 BINARY TREES not only illustrate verb phrases but they can also show the properties shared by all types of different phrases. Binary trees are a simpler but richer method of description.
++++
4 Phrases "crystalize" around specific words. The BINARY SENTENCE TREE shows these words in the center branch of a particular phrase tree. The left branch, if there is one, is the SPECIFIER. The right branch, again if there is one, contains the COMPLEMENTS.

An example of a NOUN PHRASE:



++++
5 The Xs in an X-BAR TREE can stand for anything. The X-bar tree is generalized so that any phrase, noun, verb and prepositional phrases, can be described. Here is an X-BAR TREE and a NOUN PHRASE TREE for comparison.



++++
6 An INFLECTION PHRASE adds tense information to a sentence by "inflecting on the verb". As an example, an "-ed" added to the verb indicates past tense just as the word "will" indicates future tense.
++++
7 The COMPLEMENTIZER PHRASE can be used as the highest or largest phrase type. The complementizer

```
  phrase (CP) is able to encompass and describe
  sentence-like phrases, such as inflection phrases.
  CP phrases are diagrammed in X-bar trees the same way
  as all other types of phrases.
****
```

File p3c28.exq

```
^1
1 t Near misses are used in sentence pairs to pinpoint
    language constraints.
^2
2 m A prepositional phrase
    a.  Includes a noun phrase.
    b.  Is a group of words which includes a preposition
          such as "at", "of" or "to".
    c.  Can be part of a verb phrase.
    d.  All of the above.
^3
3 m Many linguists prefer using binary trees to describe
    sentence parts because
    a.  They are able to show the properties common to
          many phrase types.
    b.  They are a simpler way to explain phrase
          structures.
    c.  Both a and b.
    d.  Neither a nor b.
^4
4 m A specifier
    a.  Is found on the left side of a binary sentence
          tree.
    b.  Can be found with verbs, nouns and prepositions.
    c.  May or may not be present in a sentence.
    d.  All of the above.
^5
5 t In an X-bar binary tree, the X may be replaced with
    a noun, verb or preposition.
^6
6 y In the sentence, "Bill moved the brick.", the
    inflection is the "-ed" on the verb "move".
^7
7 t A complementizer phrase, when diagrammed in an X-bar
    tree, can only have a complement associated and not
    a specifier.
****
```

File p3c28.exa

```
1 true
2 d
3 c
4 d
5 true
6 yes
7 false
****
```

Chapter 29

File p3c29.int

```
-CHAPTER 29
-Responding To
-Questions and Commands

     In Chapter 29 you will learn how WORD-ORDER
REGULARITY can be used to turn English questions or
commands into RELATIONAL DATABASE COMMANDS.  You will
also learn about SYNTACTIC TRANSITION-NET GRAMMAR,
NESTED-BOX DIAGRAMS and SEMANTIC TRANSITION-TREE
GRAMMARS.
****
```

File p3c29.q

```
^1
1 m  A syntactic transition-net is
     a.  Used only to identify noun phrases.
```

```
     b.  Used to test word groupings to determine if
           they are valid sentences.
     c.  Made up of a sentence net and a set of
           supporting nets.
     d.  Answers b and c.
%1
2 y A group of words is a valid sentence if the sentence
    net and all of the supporting nets and all of the
    their links are traversed.
%2
^2
3 t A nested-box diagram, for a particular group of
    words, graphically shows the path followed through
    a grammar to prove or disprove that it is a valid
    sentence.
%3
^3
4 t Relational databases hold information in tables of
    rows and columns.  The information is retrieved by
    printing a specific table.
%4
^5
5 m A semantic transition-tree grammar
    a.  Has some link transitions with specific words.
    b.  Has some link transitions specifying semantic
          phrases.
    c.  Has nodes with only one input.
    d.  All of the above.
****
```

File p3c29.a

```
1 d
2 no
3 true
4 false
5 d
****
```

File p3c29.m

```
1     A GRAMMAR is viewed, in AI, as a representation.
   It is a set of conventions for "capturing linguistic
   knowledge".  A SYNTACTIC TRANSITION-NET GRAMMAR is
   a sentence net and a set of supporting nets.
++++
2     A NESTED-BOX DIAGRAM is used to graphically
  illustrate what takes place as a route is traveled
  through a grammar.  Each box represents a link in the
  syntactic transition-net.  As each link is accepted
  or rejected, the action is so noted in the box.
++++
3     Some sentences can be translated into RELATIONAL
  DATABASE COMMANDS.  Relational databases hold infor-
  mation in tables of rows and columns.
++++
4     A SEMANTIC TRANSITION-TREE GRAMMAR uses English
  sentences to retrieve database data.  Semantic trans-
  ition-trees are different from syntactic transition-
  nets in several ways.
****
```

File p3c29.exm

```
1     A SYNTACTIC TRANSITION-NET is used to determine
   if a group of words is a valid sentence.  The process
   involves starting with the sentence net then moving
   to the noun-phrase net, the verb-phrase net and the
   prepositional-phrase net as each link in the sentence
   and supporting nets indicate.
++++
2     A group of words may be a valid sentence without
   traversing all of the supportive nets.  A sentence,
   for instance, may not have a prepositional phrase or
   an adjective.  The sentence, "The goat runs." is
   such a sentence.
++++
3     The NESTED-BOX DIAGRAM shows each link followed
   in a syntactic transition-net grammar.  Each link
   has its own box with a description of the words
   "consumed" or the indication of failure if the
```

```
   link did not apply.  The boxes are nested according
   to the structure of the surrounding phrase.
++++
4     Information is retrieved from a RELATIONAL DATA-
   BASE by using commands to select certain rows and
   columns.  This retrieved information can be combined
   with other selected information or limited further
   by combining commands.
++++
5     SEMANTIC TRANSITION-TREES have some link labels
   containing specific words (semantic) rather than the
   syntactic labels ("noun") found in SYNTACTIC TRANS-
   ITION NETS.  The nodes of a transition-tree always
   have a single input unlike the transition-net which
   can have multiple inputs.
****
```

File p3c29.exq

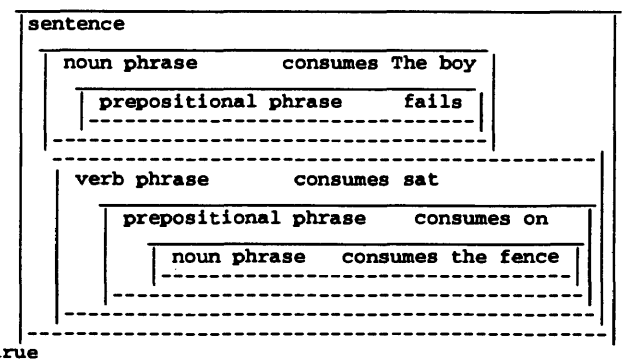
```
^1
1 y  Using this simple grammar:

       Noun phrase  verb phrase
     0------------>0------------>end
     NP determiner   noun       __
     0------------>0------>end   |
                           |<--
     VP verb     noun phrase
     0------->0------------->end
     PP preposition    noun phrase
     0-------------->0-------------->end

     Is this a sentence?
     "The beautiful weather turned bitter cold."
^2
2 y A sentence is proven valid when the sentence net
    is completely traversed but not necessarily all
    of the supportive phrase nets have been traversed.
^3
3 m The sentence:
      "The boy sat on the wooden box."
    has a nested-box diagram of _____ boxes nested
    inside of each other.
    a. 3    b. 6    c. 4    d. 5
^4
4 t A set of objects can be obtained from a relational
    database by using the SELECT and PROJECT commands.
^5
5 t A semantic transition-tree contains specific words
    and phrases rather than syntactic phrases and words.
****
```

File p3c29.exa

```
1 yes
2 yes
3 b
####
                "The boy sat on the fence."
```




```
4 true
5 true
****
```